Grafiek en geluid voor de MSX-computers

MSXB2

DATA BECKER NEDERLANDS*

Lüers

Grafiek en geluid voor de MSX-computers

MSX Bibliotheek 2

DATA BECKER NEDERLANDS*

Grafiek en geluid voor de MSX-computers

Lüers

Grafiek en geluid voor de MSX-computers

MSX Bibliotheek 2

DATA BECKER
NEDERLANDS*

Oorspronkelijke titel: MSX Grafik & Sound

Omslagontwerp: Martin van Keulen
Vertaling: ProCread, Groningen
ISBN 90 229 3358 x
D/1986/0939/54

Boeken en programma's van

DATA BECKER
NEDERLANDS*

worden in de handel gebracht door;
A.W. Bruna & Zoons Uitgeversmij b.v.,
Postbus 8411, 3503 RK Utrecht
en
A.W. Bruna & Zoon n.v.,
Antwerpsesteenweg 29a, 2630 Aartselaar.

INHOUDSOPGAVE

Voorwoord

Tot nu toe was het begrip 'uitwisselbaar' (of 'compatible') alleen van toepassing op computers uit de hogere prijsklassen. Wie een homecomputer van een bepaald merk kocht, hoefde er niet op te rekenen dat hij met randapparatuur, firm- en software van andere merken uit de voeten kon. Zelfs binnen 1 merk was overeenstemming tussen de verschillende onderdelen vaak ver te zoeken. Zodra er een nieuw model op de markt werd gebracht, 'vergat' de fabrikant alles wat daaraan voorafging en kon de bezitter van het als bij toverslag verouderde model weer van voren af aan beginnen. Op korte termijn natuurlijk aantrekkelijk voor de fabrikant, maar op de lange duur leidt zo'n politiek rechtstreeks naar de ondergang.

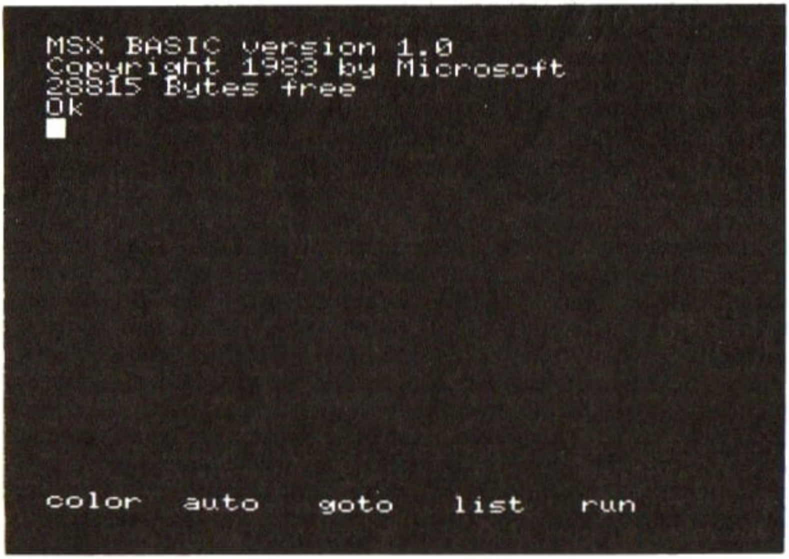
Enkele Japanse firma's hebben dit als eerste beseft. Omdat zij later op de computermarkt verschenen dan de Amerikanen en de Europeanen moesten ze gezamenlijk iets bijzonders bedenken om hun concurrenten te kunnen aftroeven. Dat werd de MSX-norm. De Japanse fabrikanten zijn overeengekomen dat niet alleen programma's (software) maar ook apparatuuraansluitingen (hardware) uitwisselbaar moeten zijn. Uitgangspunt van deze norm is de meest krachtige BASIC-versie, ontworpen door de gerenommeerde Amerikaanse softwarefirma Microsoft (MSX is een afkorting van **M**icro**S**oft e**X**tended BASIC). De MSX-standaard houdt ook in dat de chips voor grafiek en geluid in principe gelijk zijn. Elke MSX heeft (voorzover nu te overzien) standaard een Centronics interface. Daardoor is elk CP/M-programma voor de volle 100% bruikbaar. Maar een van de belangrijkste punten is wel de compatibiliteit tussen MSX-DOS en MS-DOS (beide van Microsoft). Dat betekent in de praktijk dat MS-DOS-schijfjes van het werk mee naar huis kunnen worden genomen om er op de MSX mee verder te werken. Het enige probleem is het diskette-formaat. Dat kan 5,25 of 3,5 inch zijn. Als dat niet overeenkomt, houdt natuurlijk alles op. Een hele hoop IBM machines en IBM klonen werkt met 5,25 floppies, terwijl de duurdere, maar duurzamere 3,5 inch bij de eerste generatie MSX machines favoriet is. Maar daar komt verandering in: de nieuwste IBM computers zijn uitgerust met drives van dit nieuwe type. De conclusie luidt dat de lijn kantoor-thuis in computerland steeds belangrijker wordt, hetgeen op de standaardisering een gunstig effect heeft.

U zult het met ons eens zijn dat deze uitgangspunten op den duur tot succes moeten leiden. Daarvoor zijn nog twee argumenten te geven.

- Omdat de MSX-fabrikanten afkomstig zijn uit de

Hifi/video-branche, is de tijd niet ver meer dat homecomputers kunnen worden aangesloten op videorecorder, compact disc of bandrecorder. Het voorvoegsel 'home' krijgt dan veel meer betekenis.
- Niet alleen de toepassingsmogelijkheden spreken tot de verbeelding, ook de technische gegevens mogen er zijn: uitbreidingsmogelijkheid tot 256 Kbyte RAM, 32 sprites, 16 kleuren, kleurengrafiek met hoog oplossend vermogen (256*192 pixels) en standaard een driestemmige soundprocessor.

We zullen moeten afwachten hoe de computermarkt zich gedraagt. Bedrijven uit Japan (het best vertegenwoordigd met meer dan 20 firma's, waaronder gerenommeerde namen als Sony, Panasonic en JVC), Korea (onder andere Goldstar) en Europa (waaronder natuurlijk ons eigen Philips) hopen dat in 1986 het merendeel van alle verkochte homecomputers de afkorting MSX draagt. Dat hopen wij ook.



Aan de lezer/programmeur.

DATA BECKER NEDERLANDS* heet u hartelijk welkom in de internationale kring van MSX-gebruikers. Uit de achterflap en de inhoudsopgave van dit boek heeft u al ongeveer kunnen opmaken wat u kunt verwachten. Het was niet gemakkelijk een boek te schrijven dat zowel de beginner als de prof genoeg te bieden heeft. Ik heb de gulden middenweg bewandeld door bijvoorbeeld bij de beschrijving van bijna alle MSX-instructies aanwijzingen over de systeemvariabelen te geven. Die zijn voor een beginner interessant maar voor een professional onontbeerlijk.

Ik hoop dat dit boek u zal bevallen en dat het als naslagwerk bij uw MSX goede diensten kan doen.

Met vriendelijke groet,

Rainer Lüers

1 Basisinstructies in MSX-BASIC

1.1 SCREEN

De programma's SCREEN 1-10 staan aan het einde van deze paragraaf.

SCREEN is een centrale instructie in MSX-BASIC. Hiermee kunt u niet alleen heen en weer springen tussen verschillende tekst- en grafische schermen, maar ook een veel breder en gevarieerder gebruik maken van randapparatuur. Deze veelzijdigheid is te danken aan het grote aantal parameters bij de instructie. Het algemene formaat van SCREEN is:

```
SCREEN A,B,C,D,E
```

Als u bijvoorbeeld alleen de gegevens A en E nodig heeft, kunt u de overige parameters gewoon overslaan. De komma's moeten blijven staan:

```
SCREEN A,,,,E
```

parameter A

Dit is de belangrijkste parameter: hiermee kiest u een tekstscherm (0 en 1) of een grafisch scherm (2 en 3).

SCREEN 0	alleen tekst, 1-40 tekens op 24 regels
SCREEN 1	tekst in 24 regels van maximaal 32 tekens, sprites mogelijk
SCREEN 2	grafiek met hoog oplossend vermogen (256*192 punten), sprites mogelijk
SCREEN 3	grafiek met oplossend vermogen van 64*48 punten, sprites mogelijk

Daarnaast is het mogelijk op een tekstscherm grafische voorstellingen te zetten en op een grafisch scherm tekst. Deze mogelijkheid behoeft nadere uitleg.

Grafiek op een tekstscherm

Met behulp van de grafische tekens (zoals rechthoekjes) en wat fantasie kunt u figuurtjes tekenen. Om een idee te krijgen van de mogelijkheden staan twee wegen open: tegelijk de toets GRAPH en

een willekeurige lettertoets indrukken of gebruik maken van de functie CHR$. In beide gevallen krijgt u de grafische tekens op het beeldscherm te zien; in dit boek kiezen we meestal de laatste mogelijkheid omdat de grafische tekens op die manier volledig identificeerbaar zijn. Als u de grafische karakterset van de MSX wilt bekijken, voer dan voorbeeld SCREEN 1 in (zie listing). U hoeft zich niet te beperken tot deze set; u kunt zelf letters, cijfers en ieder ander willekeurig teken definiëren. Daartoe roept u in het geheugen het BASE-adres aan waar het karakter begint.

```
PRINT BASE(2)

resultaat: 2048
```

Normaal gesproken is bijna het hele beeldscherm gevuld met spaties, die net als ieder ander teken uit 8 lijnen bestaan. Om de spatie (CHR$(32)) een ander uiterlijk te geven, kunt u de volgende truc toepassen.

```
PRINT VPEEK(BASE(2)+(32*8)+1)
```

Met de instructie VPEEK kunt u in het video-RAM precies de geheugenplaats bekijken waarin u geïnteresseerd bent. De waarde 0 verraadt dat een spatie uit tenminste een lege lijn (=0) bestaat. Wat zou er gebeuren als we deze geheugenplaats veranderen?

```
VPOKE BASE(2)+(32*8)+1,255
```

Gelijk na invoer van deze regel verandert een groot deel van het beeldscherm. Op de plaats van de spatie staat in het geheugen nu de code voor een witte streep (255). Op een tekstscherm kunnen 256 verschillende karakters worden weergegeven en daarom verandert de nieuw ingevoerde code meteen iedere spatie (een van de 256 karakters). Dit storende effect kan teniet worden gedaan door met VPOKE de oude waarde te herstellen. Hetzelfde effect bereikt u op wat minder subtiele wijze met SCREEN (A); alle oorspronkelijke letter- en tekenfuncties worden dan weer geladen. Als we ervan afzien tegelijk tekst en grafiek te willen afbeelden (dus binnen SCREEN 0 en SCREEN 1), is het bijna onmogelijk om op de boven beschreven manier op een stukje van het tekstscherm (in totaal 256*192 punten) grafische plaatjes te maken. De plaatjes die het programma SCREEN 2 creëert zijn toevalstreffers. Met de instructie SCREEN 0 komt alles na die drastische verandering weer op zijn pootjes terecht. U moet die blind intypen omdat er op het scherm alleen maar nonsens komt te staan dankzij het programma SCREEN 2. Wis daarna met CTRL-E de rest van de regel. U heeft nu een idee van de mogelijkheden van het tekstscherm SCREEN 0. Grafische instructies als CIRCLE, PAINT en LINE zijn op

tekstschermen niet gewenst. Roept u ze toch aan, dan leidt dat onherroepelijk tot een foutmelding: Illegal function call.

Tekst op een grafisch scherm

Bij de MSX is een eenvoudige PRINT-instructie niet genoeg om tekst op een grafisch scherm te krijgen (zie voorbeeld SCREEN 3). Schrijven op een grafisch scherm is wel mogelijk als u eerst een kanaal opent met OPEN. Daarna stuurt u de instructie PRINT # direct naar dat kanaal (zie programma SCREEN 4). In regel 30 van dit programma geeft u aan via welk kanaal de uitvoer van gegevens naar het grafische scherm moet plaatsvinden. Nu heeft u in principe de mogelijkheid om op grafische manier, dus pixel voor pixel, een tweede tekst over de eerste heen te schrijven. Er ontstaan alleen problemen als u tegelijkertijd verschillende kleuren wilt gebruiken. Met de instructie DRAW kunt u in een grafisch scherm een tekst pixel voor pixel op de gewenste plaats zetten, analoog aan verplaatsing van de cursor met LOCATE. Als u het programma SCREEN 5 invoert, verschijnt er een sinuscurve op het scherm, bestaand uit de letters MSX. In de modi SCREEN 2 en SCREEN 3 kunt u kleurengrafiek samen met tekst afbeelden. Let er bij SCREEN 3 op dat u de grootte van de af te beelden tekens van tevoren goed schat (in verband met de beschikbare ruimte). Het coördinatenbereik is in beide modi hetzelfde. Oplettende lezers die de instructie DRAW al kennen, hebben nu misschien twee vragen.

- Kunnen tekens op een andere schaal worden weergegeven met behulp van de instructie DRAW "S"? Antwoord: nee, daarvoor moet u een nieuwe karakterset maken met DRAW. In voorbeeld SCREEN 5 heeft DRAW alleen betrekking op de plaatsing van de grafische cursor en niet op de vorm van de uitgevoerde letters en tekens. Toch bestaat de mogelijkheid om tekst vergroot en/of gedraaid weer te geven. Met de instructie POINT kunnen we pixels van het scherm lezen en met PSET de veranderde weergave direct op het scherm uitvoeren (zie voorbeeld SCREEN 6).
- Met SCREEN 0 schakelen we over naar de standaard karakterset. Die set moet dus nog in het ROM aanwezig zijn. Antwoord: dat klopt. Als u voorbeeld SCREEN 7 invoert, verschijnt dat ROM-gedeelte binair weergegeven op het scherm. Dezelfde karakterset (gekopieerd uit het genoemde ROM-bereik) bevindt zich bij SCREEN 0 in het video-RAM.

Weergave van tekst in SCREEN 2 (grafisch scherm) lijkt alleen uiterlijk op een tekstscherm. Op een grafisch scherm kunt u geen invloed uitoefenen op het aantal tekens per regel (32), tenzij u letter voor letter tekent met DRAW "B M". Op een tekstscherm kan dat wel (met WIDTH). Bovendien scrollt het scherm niet als het

einde van de laatste schermregel (regel 24) is bereikt. In plaats daarvan beschrijft de MSX het scherm opnieuw; niet door het oude te wissen maar door er overheen te tekenen. Het is bovendien erg lastig tekst van een grafisch scherm te verwijderen zonder het hele scherm te wissen. Er zijn drie manieren om langs een omweg tekst te laten verdwijnen. Ten eerste kunt u een rechthoek ter grootte van de te wissen tekst tekenen en deze vullen met de achtergrondkleur (functie LINE ...,,BF). Ten tweede kunt u precies dezelfde tekst nog eens intypen op dezelfde plaats, maar dan in de achtergrondkleur. Tenslotte zou u met VPOKE in het video-RAM de tekst adres voor adres weer in de achtergrondkleur kunnen zetten. Bekijk tot slot voorbeeld SCREEN 9. Daarin wordt tekst met behulp van een gevulde rechthoek gewist, opnieuw op het scherm gezet, weer gewist, etc. Dat geeft een aardig effect.

parameter B

Parameter B heeft betrekking op het uiterlijk van sprites op het scherm. Als u alleen deze parameter nodig heeft en A niet, laat dan alleen de komma staan (bijvoorbeeld SCREEN ,1). Er zijn vier mogelijkheden:

SCREEN ,0	sprite-formaat 8* 8	normale pixelgrootte
SCREEN ,1	sprite-formaat 8* 8	vergrote pixels
SCREEN ,2	sprite-formaat 16*16	normale pixelgrootte
SCREEN ,3	sprite-formaat 16*16	vergrote pixels

Op het scherm heeft u steeds maar 1 sprite-formaat tegelijk ter beschikking. Hou daar rekening mee als u gaat programmeren. Om de mogelijkheden van tevoren goed te kunnen bekijken, kunt u met de instructie (c.q. variabeletoewijzing) SPRITE$ eerst een 16*16 sprite definiëren. De informatie die de MSX bij het gekozen formaat niet nodig heeft, komt niet op het scherm. Van een 16*16 sprite bijvoorbeeld ziet u in SCREEN 0 alleen de linkerbovenhoek. Bij omschakeling naar een ander formaat krijgt het spritegedeelte van het VRAM automatisch een nieuwe indeling. Alle sprites op het scherm en sprite-definities in het VRAM worden dan gewist. Voorbeeld SCREEN 10 laat de vier sprite-formaten na elkaar zien. Voor de volledigheid merken we nog op dat bij elk sprite-formaat maximaal 32 verschillende sprites tegelijk op het scherm kunnen staan. In het VRAM ligt dat anders. Er kunnen 256 sprites van het formaat 8*8 worden opgeslagen. 16*16 sprites bestaan in feite uit 2*2=4 sprites van het type 8*8. Er passen dus 256:4 = 64 van deze sprites in de sprite generator table in het VRAM, die een exact begrensd bereik heeft.

parameter C

Het kan handig zijn om te weten of een toets al dan niet is ingedrukt. Deze parameter zorgt voor een zachte klik bij iedere toetsdruk. Een fout klinkt heel anders: een penetrante pieptoon. Mocht het geklik u op de zenuwen gaan werken dan kunt u dat met SCREEN ,,0 laten ophouden. Om de klik weer terug te krijgen kiest u voor parameter C een getal tussen de 1 en 255.

parameter D

U kunt programma's opslaan op band of schijf. Met een cassetterecorder duurt dat bij een wat langer spelprogramma (zoals achter in dit boek) wel een paar minuten. Parameter D heeft twee opties. Als u apparatuur en bandjes van een normale kwaliteit heeft, kunt u het beste SCREEN ,,,1 kiezen. De MSX slaat gegevens dan op met een snelheid van 1200 Baud (= 1200 bits per seconde = ongeveer 150 karakters per seconde). Bij gebruik van een goede recorder en dito kwaliteit ferro-cassettes zorgt SCREEN ,,,2 ervoor dat de MSX de gegevens verwerkt met dubbele snelheid (2400 Baud).

parameter E

Deze parameter heeft betrekking op printers. Als u een printer heeft die niet aan de MSX-norm voldoet, kunnen problemen ontstaan. De letters en de andere karakters tot en met ASCII-code 127 kunnen op elk type printer worden uitgevoerd. Daarnaast heeft de MSX een groot aantal bijzondere en grafische tekens (zoals bijvoorbeeld de trema). Voor een MSX printer is dat natuurlijk geen probleem. Kies hiervoor optie SCREEN ,,,,0. Is er sprake van een ander type printer, neem dan voor parameter E een andere waarde (1-155). Tekens vanaf CHR$(128) worden dan uitgevoerd als spaties en de printer kan er dus niets aan verpesten. Als u dat wenst kunt u de speciale tekens dan makkelijk met de hand intekenen.

SCREEN 1
```
10 COLOR 1,15
20 SCREEN 1
30 FOR N=192 TO 255
40 PRINT CHR$(N);
50 NEXT N
```

SCREEN 2
```
10 COLOR 1,15
20 SCREEN 0
30 FOR N=0 TO 959
40 PRINT CHR$(INT(RND(1)*223)+32);
50 NEXT N
60 FOR N=2048 TO 4095
70 VPOKE N,INT(RND(1)*255)
80 NEXT N
```

SCREEN 3
```
10 COLOR 1,15
20 SCREEN 2
30 PRINT "MSX-computer"
40 GOTO 40
```

SCREEN 4
```
10 COLOR 1,15
20 SCREEN 2
30 OPEN "grp:" FOR OUTPUT AS #1
40 PRINT #1, "MSX-computer"
50 CLOSE
60 GOTO 60
```

SCREEN 5
```
10 COLOR 1,15
20 SCREEN 2
30 OPEN "grp:" FOR OUTPUT AS #1
40 FOR N=1 TO 255 STEP 3
50 Y=SIN(N/20)*50+100
60 DRAW "bm =n;,=y;"
70 PRINT #1,"MSX";
80 NEXT N
90 CLOSE
100 GOTO 100
```

SCREEN 6
```
10 COLOR 1,15
20 SCREEN 2
30 OPEN"grp:"FOR OUTPUT AS #1
40 DRAW "bm 0,0"
50 PRINT #1,"MSX"
60 FOR N=0 TO 23
70 FOR M=0 TO 7
80 A=POINT(N,M)
90 PSET(N+50,M+50+Z),A
100 Z=Z+1
110 PSET(N+50,M+50+Z),A
```

```
        120 NEXT M
        130 Z=0:NEXTN
        140 GOTO 140

SCREEN 7  10 COLOR 1,15
          20 SCREEN 0
          30 WIDTH 40
          40 FOR N=&H1BBF TO &H1BBF+2047
          50 PRINT STRING$(8-LEN(BIN$(PEEK(N))),"0");
          BIN$(PEEK(N))
          60 NEXT N

SCREEN 8  10 COLOR 1,15
          20 SCREEN 0
          30 WIDTH 40
          40 FOR N=2048 TO 2048+2047
          50 PRINT STRING$(8-LEN(BIN$(VPEEK(N))),"0");
          BIN$(VPEEK(N))
          60 NEXT N

SCREEN 9  10 COLOR 1,15
          20 SCREEN 2
          30 OPEN"grp:"  FOR OUTPUT AS #1
          40 DRAW "bm 128,96"
          50 PRINT #1,"MSX"
          60 FOR N=1 TO 200
          70 NEXT N
          80 LINE(128,96)-(152,104),15,BF
          90 FOR N=1 TO 200
          100 NEXT N
          110 GOTO 40

SCREEN 10 10 COLOR  1,15
          20 OPEN "grp:" FOR OUTPUT AS #1
          30 FOR N=0 TO 3
          40 SCREEN 2,N
          50 DRAW "bm 50,50"
          60 SPRITE$(1)=STRING$(32,"U")
          70 PRINT #1,"grootte";N
          80 PUT SPRITE 1,(128,96)
          90 FOR M=1 TO 1000
          100 NEXT M
          110 NEXT N
          120 GOTO 30
```

1.2 OPEN

De programma's OPEN 1-10 staan aan het einde van deze paragraaf.

Met de instructie OPEN kunt u kanalen voor gegevens openen, waarbij u de keuze hebt uit

CAS	cassetterecorder
LPT	printer (line printer)
CRT	tekstscherm (cathode ray tube)
GRP	tekst op een grafisch scherm
1/2	diskdrive(s)

Dit boek gaat over grafiek en geluid, daarom bespreken we alleen de uitvoermogelijkheden die daarbij horen, te weten LPT, CRT en GRP.

OPEN "LPT"/OPEN "CRT"

In de directe modus zult u deze instructies zelden gebruiken want in MSX-BASIC bestaan daarvoor een paar uitstekende instructies:

PRINT "..."	uitvoer van tekst op het scherm
LPRINT "..."	idem, op een printer
LIST	uitvoer van een listing op het scherm
LLIST	idem, op een printer

Voor alle duidelijkheid volgt hier de volledige procedure voor uitvoer van tekst met de instructie OPEN.

```
OPEN "LPT:" FOR OUTPUT AS #1
PRINT #1, "MSX is niet niks"
CLOSE #1
```

In MSX-BASIC:

```
LPRINT "MSX is niet niks"
```

Waarom moeilijk als het makkelijk kan? Heel simpel: de methode met OPEN heeft het voordeel dat de uitvoer de ene keer naar kanaal 1 wordt gestuurd en de volgende keer naar kanaal 2 (zie voorbeeld 1). U zou door nog meer kanalen te openen (3 en 4) de uitvoer ook naar een cassetterecorder (CAS), een diskdrive (1) of een grafisch scherm (GRP) kunnen sturen. We kunnen de uitvoerapparaten aanspreken met de getallen 1-4. Het is dus vrij eenvoudig om de hele uitvoer in een programma naar de printer te sturen in plaats van naar het scherm. Dat bespaart heel wat tijd en ruimte. Wie wel eens in een lang programma alle PRINT-instructies heeft moeten vervangen door LPRINT zal blij zijn dat hij met OPEN maar 1

variabelewaarde hoeft te veranderen. Er is nog een verschil tussen uitvoer met PRINT of LPRINT enerzijds en met OPEN "CRT:" of OPEN "LPT:" anderzijds. Dat komt tot uiting bij geformateerde uitvoer (uitvoer in kolommen voor bijvoorbeeld tabellen). Voor een geformateerde uitvoer moet u aan de tekst achter (L)PRINT een komma toevoegen. Om de 14 posities wordt dan een TAB (tabulatorstop) gezet en daar wordt de uitvoer steeds voortgezet. Mocht een stukje uitvoer langer zijn dan het resterende aantal positites op een regel dan wordt het in zijn geheel afgedrukt op de eerste TAB van de volgende regel. Bij OPEN "LPT:" of OPEN "CRT:" gevolgd door een komma gaat de uitvoer verder na 14 posities zonder rekening te houden met regeleinde of schermrand (bij OPEN "LPT:" zit het nog iets anders). De voorbeelden 2-5 tonen achtereenvolgens het resultaat van PRINT, OPEN "CRT:", LPRINT en OPEN "LPT:". Geformateerde uitvoer is dus niet mogelijk als u een kanaal hebt geopend met OPEN.

OPEN "GRP:"

Er is geen wezenlijk verschil tussen uitvoer via een gegevenskanaal of via LPRINT/LLIST. Bij de grafiekmodi ligt dat anders. Probeer maar eens met de instructie PRINT tekst op een grafisch scherm te krijgen zonder met OPEN een kanaal te openen (voorbeeld 6). Hoe mooi en doorzichtig het programma ook is, er komt niets van terecht. Nu proberen we het met OPEN "GRP:" (voorbeeld 7). Het resultaat is perfect en linksboven is de tekst duidelijk zichtbaar. We zeiden al dat werken op een grafiekscherm volgens heel andere regels verloopt dan op een tekstscherm. Tekstuitvoer betekent eigenlijk niets anders dan letter voor letter een nieuwe tekeing van 8*8 punten. Er zijn dan ook twee invoermogelijkheden:

- regel voor regel, met de normale teken- en regelafstand (zoals in SCREEN 1), waarbij ruimte is voor 32*24 tekens (zie voorbeeld 8)

- punt voor punt de beeldschermposities aanroepen, zodat de tekst precies op de plaats komt waar we deze willen hebben (het 32*24-matrixveld hebben we dan niet meer nodig)

Op een grafiekscherm hebben we niet de beschikking over de instructie LOCATE zodat we de cursor blind moeten verplaatsen met DRAW "B M". Om de kleur van een uitgevoerde tekst te veranderen moet u ondanks het verfijnde karakter van de instructie DRAW toch nog COLOR invoeren. Verwacht hier niet teveel van. De uitgevoerde tekens kunnen niet worden gedraaid (DRAW "A") en ook niet vergroot (DRAW "S").

Een ander verschil tussen tekst- en grafiekscherm blijkt bij invoer van tekst op de onderste regel van een grafiekscherm. Als zo'n scherm vol is, scrollt het niet omhoog maar wordt het opnieuw

beschreven, te beginnen linksboven. Er zijn twee manieren om een grafiekscherm te wissen: ten eerste volledig wissen met CLS en ten tweede gedeeltelijk wissen door eroverheen schrijven met de achtergrondkleur. Dat laatste is vergelijkbaar met de correctie-toets van een elektrische typemachine (zie voorbeeld 10). De nieuwe tekst moet volledig overeenkomen met de oude, want anders blijven er stukjes over. Verander regel 90 maar eens in:

```
90 PRINT //1, "MSX cantuter"
```

Na de start van het veranderde programma met RUN blijven een paar restjes van het woord computer op het scherm staan omdat het woord 'cantuter' in regel 90 niet geheel overeenkomt met het woord 'computer' in regel 60.

OPEN 1

```
10 COLOR 1,15
20 SCREEN 0
30 MAXFILES=2
40 OPEN "crt:" FOR OUTPUT AS #1
50 OPEN "lpt:" FOR OUTPUT AS #2
60 FOR N=1 TO 2
70 PRINT #N,"MSX-computers zijn GEWELDIG"
80 NEXT N
90 CLOSE
```

OPEN 2

```
10 COLOR 1,15
20 SCREEN 0
30 FOR N=1 TO 20
40 PRINT"fantastische MSX",
50 NEXT N
```

OPEN 3

```
10 COLOR 1,15
20 SCREEN 0
30 OPEN "CRT:" FOR OUTPUT AS #1
40 FOR N=1 TO 20
50 PRINT #1, "fantastische MSX",
60 NEXT N
70 CLOSE
```

OPEN 4

```
10 FOR N=1 TO 20
20 LPRINT "fantastische MSX",
30 NEXT N
```

OPEN 5

```
10 OPEN "LPT:" FOR OUTPUT AS #1
20 FOR N=1 TO 20
30 PRINT #1, "MSX is fantastisch",
40 NEXT N
50 CLOSE
```

OPEN 6

```
10 COLOR 1,15
20 SCREEN 2
30 PRINT"MSX en grafiek"
40 GOTO 40
```

OPEN 7

```
10 COLOR 1,15
20 SCREEN 2
30 OPEN "grp:" FOR OUTPUT AS #1
40 PRINT #1,"MSX en grafiek"
50 GOTO 50
```

OPEN 8

```
10 COLOR 1,15
20 SCREEN 2
30 OPEN "grp:" FOR OUTPUT AS #1
40 FOR N=1 TO 100
```

```
50 PRINT #1,"MSX ";
60 NEXT N
70 CLOSE
80 GOTO 80
```

OPEN 9

```
10 COLOR 1,15
20 SCREEN 2
30 OPEN "grp:" FOR OUTPUT AS #1
40 FOR N=1 TO 255 STEP 3
50 M=SIN(N/20)*50+100
60 DRAW "bm =n;,=m;"
70 PRINT #1, "MSX"
80 NEXT N
90 CLOSE
100 GOTO 100
```

OPEN 10

```
10 COLOR  1,15
20 SCREEN 2
30 OPEN "grp:" FOR OUTPUT AS #1
40 DRAW "bm 128,96"
50 COLOR 1
60 PRINT #1,"MSX-computer"
70 DRAW "bm 128,96"
80 COLOR 15
90 PRINT #1,"MSX-computer"
100 GOTO 40
```

1.3 VDP

De toewijzing van waarden bij deze instructie moet in binaire vorm plaatsvinden. We lopen de daarvoor geldende regels nog even langs. Een byte bestaat uit 8 bits. Een bit heeft de waarde 0 of 1. Iedere bit staat voor een macht van 2:

bit	7	6	5	4	3	2	1	0
macht	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
=	128	64	32	16	8	4	2	1

De waarde van een byte verkrijgen we door de 8 bits op te tellen. Is een bit 1, dan doet de betreffende macht mee in de optelling. Zo kunnen de getallen van 1 tot en met 255 binair worden weergegeven. Een variabele krijgt een binaire waarde door voor het getal de tekencombinatie &B te zetten. De toekenning van het binaire getal 00100100 aan de variabele A ziet er zo uit:

```
A = &B00100100
```

De instructie VDP (van video display processor) is een vorm van variabeletoewijzing. Het algemene formaat is VDP(n), waarbij n een getal tussen 0 en 8 is en het gewenste register aangeeft. Met VDP spreken we de processor direct aan en geven aan welk register invoer moet krijgen of moet worden gelezen. Pas daar goed mee op tot u de opbouw van het video-RAM doorgrondt. VDP is een instructie die diep in de MSX doordringt. Als er iets verkeerd gaat, is het gelijk goed mis: in het ergste geval raakt u door verkeerde programmering het programma kwijt of kunt u het niet meer op het beeldscherm krijgen. De computer zelf kan er geen schade door oplopen. U kunt de registers 0-7 lezen met PRINT VDP(n) en er waarden in zetten met VDP(n)=m. Register 8 (status register) kunt u alleen lezen. Om de actuele inhoud van de registers 0-7 nog op een andere manier te kunnen lezen, is die ook opgeslagen op de geheugenplaatsen &HF3DF-&HF3E6. Ook de inhoud van register 8 heeft een kopie in het RAM en wel op &HF3E7. Bij de bespreking van de afzonderlijke registers gaan we uit van de inhoud van elk register in binaire vorm zoals die op het scherm verschijnt nadat de MSX is aangezet. De modus is dan SCREEN 0, waarbij de voorgrond wit en de achtergrond blauw is.

REGISTER 0

oproep actuele inhoud	PRINT PEEK(&HF3DF)
binair	00000000
toewijzing	VDP(0)=...

In dit register hebben alleen bit 0 en 1 een functie. Bit 2-7 doen niet mee en mogen een willekeurige waarde hebben. Zolang de MSX 1 video display processor heeft, heeft bit 0 de waarde 0. De ontwerpers hebben rekening gehouden met een tweede VDP. Zodra die in actie komt, krijgt bit 0 de waarde 1. Bit 1 geeft samen met bit 3 en bit 4 van register 1 de actuele SCREEN-modus aan; zie verder de bespreking van register 1. Hier volstaat de mededeling dat bit 1 van register 0 meestal 0 is. Alleen bij SCREEN 2 (grafiekscherm met hoog oplossend vermogen) is deze bit 1. Voorbeelden:

VDP(0)=&B000000**01**	tweede VDP ingeschakeld
VDP(0)=&B000000**10**	omschakeling naar SCREEN 2, mits de bits 3 en 4 van register 1 beide 0 zijn

REGISTER 1

oproep actuele inhoud	PRINT PEEK(&HF3DF)
binair	00000000
toewijzing	VDP(0)=...

bit 0

- Bit 0 krijgt de waarde 0 als u het normale sprite-formaat kiest. 8*8 sprites zijn dan 8*8 pixels groot. Hetzelfde geldt voor 16*16 sprites.
- Bit 0 krijgt de waarde 1 als u de sprites vergroot wilt afbeelden. Iedere punt wordt op het scherm 2*2 pixels. Sprites van respectievelijk 16*16 en 32*32 pixels zijn het resultaat.

Bit 0 van register 1 heeft dezelfde functie als de BASIC-instructie SCREEN ,B. Waarom dan moeilijk als het ook makkelijk kan? Heel simpel: als u in machinetaal programmeert, beschikt u niet over de instructie SCREEN. U werkt dan met een VDP-register en de instructies INP en OUT.

bit 1

- Bit 1 krijgt de waarde 0 als u werkt met 8*8 sprites, vergelijkbaar met SCREEN ,0. U zou hier ook SCREEN ,1 kunnen kiezen mits bit 0 van register 1 op 1 staat.
- Bit 1 krijgt de waarde 1 als u met 16*16 sprites werkt, vergelijkbaar met SCREEN ,2. Ook SCREEN ,3 is hier mogelijk mits bit 0 van register 1 op 1 staat.

We wijzen hier op een belangrijk verschil tussen de instructies VDP en SCREEN ,N: bij SCREEN worden eerder gedefinieërde sprites in het VRAM gewist terwijl dat bij VDP niet gebeurt. Het

programma in paragraaf 9.7 demonstreert de vier verschillende sprite-groottes (c.q. sprite-oplossingen).

bit 2

Deze bit heeft in dit register geen functie. Het maakt niet uit of u er een 0 of een 1 neerzet, het beeld op het scherm blijft hetzelfde.

bit 3 en bit 4

De bits 3 en 4 werken nauw samen en informeren bovendien over de status van bit 1 in register 0. Met deze drie bits kunnen we het gewenste of het actuele SCREEN bepalen.

register	0	1	1
bit	1	3	4
SCREEN 0	0	0	1
SCREEN 1	0	0	0
SCREEN 2	1	0	0
SCREEN 3	0	1	0

Met behulp van deze drie bits kunt u naar hartelust van SCREEN wisselen. Parameter A van SCREEN hebben we niet meer nodig. Als we ook de bits 0 en 1 van register 1 erbij nemen, kan ook parameter B weg en kunnen we de VDP direct aanspreken. Dat is handig als u in machinetaal programmeert, maar ook in BASIC kunt u er profijt van hebben, namelijk als u het VRAM naar eigen inzicht wilt indelen (zie de bespreking van de registers 2-6).

bit 5

Bit 5 schakelt de automatische controle bij een machine-interrupt uit (0) of aan (1). Deze functie hangt samen met de instructies ON SPRITE GOSUB en SPRITE ON/OFF/STOP. Als de MSX intern moet testen op botsingen, gebruiken we daarvoor in BASIC de instructie SPRITE ON. De computer springt dan naar een subroutine (aangegeven met ON SPRITE GOSUB) die bijvoorbeeld een explosie op het scherm regelt. Deze test op botsingen schakelt u uit met SPRITE OFF. Bit 5 in register 1 doet hetzelfde: 1 betekent SPRITE ON, 0 wil zeggen SPRITE OFF. SPRITE STOP kan niet via een VDP-register worden bestuurd.

bit 6

Normaal werkt u met voorgrondkleur, achtergrondkleur en eventueel een kaderkleur; bit 6 is dan 1. Is deze bit 0, dan wordt de voorgrondkleur gelijk aan de kaderkleur. U kunt dan op het scherm

tekst invoeren of figuren ontwerpen die pas zichtbaar worden als bit 6 gelijk is aan 1.

bit 7

Bit 7 is altijd 1 en geeft aan dat de VDP de beschikking heeft over minstens 8 Kbyte en normaliter zelfs 16 Kbyte VRAM. In de ontwerpfase van de MSX hield men nog rekening met de mogelijkheid dat de MSX maar over 4 Kbyte VRAM zou beschikken. In dat geval zou bit 7 de waarde 0 hebben.

Voorbeelden bij register 1

VDP(1)=&B**11100011**

bit	waarde	
0	1	vergrote sprites (iedere spritepunt is 2*2 pixels)
1	1	weergave van 16*16 sprites

Bit 0 en 1 komen samen overeen met BASIC-instructie SCREEN ,3

2	0	zonder betekenis
3 4	0 en 0	Als bit 1 van register 0 gelijk is aan 1, hebben ze dezelfde betekenis als SCREEN 2. Als bit 1 van register 0 gelijk is aan 0, hebben ze dezelfde betekenis als SCREEN 1.
5	1	interruptsturing voor sprite-botsing ingesteld (BASIC-instructie SPRITE ON)
6	1	normaal beeldscherm: grafiek heeft kleur van voorgrond
7	1	8 of 16 Kbyte VRAM beschikbaar

VDP(1)=&B**10001000**

bit	waarde	
0	0	weergave van sprites op ware grootte
1	0	alleen kleine sprites (8*8 pixels)
2	0	geen betekenis

3	1 en	Als bit 1 van register 0 gelijk is aan 0, is het actuele scherm SCREEN 3 (blokgrafiek).
5	0	test op spritebotsing uitgeschakeld (BASIC-instructie SPRITE OFF)
6	0	grafiek heeft kaderkleur, is dus niet zichtbaar (dat is wel het geval als bit 6=1)
7	1	8 of 16 Kbyte VRAM beschikbaar

Voordat we de registers 2-6 bespreken het volgende: de bits in deze registers zijn steeds onderdeel van een groter binair getal. Ze krijgen dus andere waarden dan ze eigenlijk zouden moeten hebben. De betreffende bits zijn vet gedrukt. We geven steeds de grootste waarde van het register aan; de kleinste waarde (alle bits zijn 0) is altijd decimaal 0. In bijlage 2 vindt u een opsomming van alle mogelijke waarden die deze registers kunnen krijgen. Verder geldt voor al deze registers dat door verandering van de relevante bits het beginadres verschuift van de table waarop het register betrekking heeft. Bij elk register geven we aan wat voor mogelijkheden die verschuiving biedt.

REGISTER 2

oproep actuele inhoud	PRINT PEEK(&HF3E1)
binair	00000000
toewijzing	VDP(2)=...

Alleen de bits 0-3 hebben een functie. Ze moeten worden geïnterpreteerd als bit 10-13. De grootste waarde heeft dit register als alle vier bits 1 zijn.

grootste waarde 00**1111**0000000000 = decimaal 15360

Als alleen bit 0 (bit 10 in deze vorm) de waarde 1 heeft, levert dat het decimale getal 1024 (2^10) op. Met de vier bits van register 2 bepalen we waar de pattern name table begint, oftewel waar de opbouw van het beeldscherm in het VRAM is opgeslagen (teken voor teken respectievelijk pixel voor pixel). Als register 2 in zijn beginstand staat, geldt voor de SCREENs het volgende.

	start	lengte
SCREEN 0	0	960
SCREEN 1	6144	768
SCREEN 2	6144	768
SCREEN 3	2048	768

Toepassing: u kunt bijvoorbeeld onder SCREEN 0 verschillende schermen opslaan in het VRAM en deze snel 'doorbladeren'. Deze mogelijkheid is nader uitgewerkt in het programma in paragraaf 9.2.

Voorbeelden bij register 2

VDP(2)=&B0100

Het begin van de pattern name table in het VRAM is adres &B00**0100**0000000000 = decimaal 5120.

VDP(2)=&B1111

Het begin van de pattern name table in het VRAM is adres &B00**1111**0000000000 = decimaal 15360.

REGISTER 3

oproep actuele inhoud	PRINT PEEK(&HF3E2)
binair	00000000
toewijzing	VDP(3)=...

In register 3 kunt u alle 8 bits een waarde geven, maar ze worden behandeld alsof het om de posities 6-13 van een getal van 16 bits gaat.

grootste waarde 00**11111111**000000 = decimaal 16320

Als alleen bit 0 is gezet (in deze vorm bit 6), betekent dat decimaal 64 (2^6). Met register 3 bepalen we waar de colour table begint, oftewel waar de informatie over de kleuren van tekst/grafiek op het beeldscherm is opgeslagen in het VRAM. In de beginstand betekent dit register het volgende voor de verschillende SCREENs:

	start	lengte
SCREEN 0	----	----
SCREEN 1	8192	32
SCREEN 2	8192	6144
SCREEN 3	0	2048

Toepassing: u kunt bijvoorbeeld op SCREEN 1 informatie opslaan over de 32 kleur-bytes en deze snel 'doorbladeren'.

Voorbeelden bij register 3

VDP(3)=&B01010101

De colour table in het VRAM begint op adres &B00**01010101**000000, decimaal 5440.

VDP(3)=&B11111111

De colour table in het VRAM begint op adres &B00**11111111**000000, decimaal 16320.

REGISTER 4

oproep actuele inhoud	PRINT PEEK(&HF3E3)
binair	00000001
toewijzing	VDP(4)=...

Alleen de bits 0-2 hebben een functie. Ze worden geïnterpreteerd als bit 11, 12 en 13.

grootste waarde 00**111**00000000000 = decimaal 14336

Als alleen bit 1 (in deze vorm bit 11) van dit register 1 is, wordt dat decimaal 2048 (2^11). Met dit register bepalen we waar de pattern generator table in het VRAM begint. Deze bevat informatie over de vorm van de tekens en van de grafiek met hoog oplossend vermogen. In de beginstand levert dit register de volgende indeling:

	start	lengte
SCREEN 0	2048	2048
SCREEN 1	0	2048
SCREEN 2	0	6144
SCREEN 3	----	----

Toepassing: u kunt bijvoorbeeld onder SCREEN 0 verschillende karaktersets in het VRAM opslaan en steeds de gewenste karakterset oproepen (zie de programma's in paragraaf 9.3 en 9.14).

Voorbeelden bij register 4

VDP(4)=&B010

De pattern generator table begint op adres &B00**010**00000000000, decimaal 4096.

VDP(4)=&B111

De pattern generator table begint op adres &B00**111**00000000000, decimaal 14336 in het VRAM.

REGISTER 5

oproep actuele inhoud	PRINT PEEK(&HF3E4)
binair	00000000
toewijzing	VDP(5)=...

De bits 0-6 hebben een functie. Ze worden geïnterpreteerd als bit 7-13.

grootste waarde &B00**1111111**0000000 = decimaal 16256

Als alleen bit 0 (in deze vorm bit 7) gelijk is aan 1 staat, betekent dat decimaal 128 (2^7). Met dit register bepalen we waar de sprite attribute table in het VRAM begint. Deze bevat informatie over de plaats waar sprites op het scherm staan (aangeduid met X- en Y-coördinaten), welke kleur ze hebben en op welk sprite-formaat in het VRAM ze betrekking hebben. In de beginstand betekent dit register het volgende voor de verschillende SCREENs:

	start	lengte
SCREEN 0	----	---
SCREEN 1	6912	128
SCREEN 2	6912	128
SCREEN 3	6912	128

Toepassing: u kunt bijvoorbeeld onder SCREEN 1 meerdere sprite-posities en sprite-kleuren opslaan in het VRAM en deze snel veranderen.

Voorbeelden bij register 5

VDP(5)=&B0101010

De sprite attribute table begint op adres &B00**0101010**0000000, decimaal 5376.

VDP(5)=&B1111111

De sprite attribute table begint op adres &B00**1111111**0000000, decimaal 16256.

REGISTER 6

oproep actuele inhoud	PRINT PEEK(&HF3E5)
binair	00000000
toewijzing	VDP(6)=...

Alleen bit 0, 1 en 2 hebben een functie. Ze worden geïnterpreteerd als bit 11-13.

grootste waarde 00**111**0000000000 = decimaal 14336

Als alleen bit 0 (hier bit 11) gelijk is aan 1, betekent dat decimaal 1024. Met dit register bepalen waar in het VRAM de sprite generator table begint. Afhankelijk van de geldende SCREEN-modus bepaalt de SGT de vorm van maximaal 256 sprites van 8*8 of maximaal 64 sprites van 16*16. In de begintoestand heeft dit register de volgende indeling:

	start	lengte
SCREEN 0	-----	----
SCREEN 1	14336	2048
SCREEN 2	14336	2048
SCREEN 3	14336	2048

Toepassing: u kunt bijvoorbeeld onder SCREEN 1 verschillende groepen (die elk tot maximaal 256 sprites van 8*8 bevatten) in het VRAM opslaan en die snel achter elkaar bekijken.

Voorbeelden bij register 6

VDP(6)=&B010

De SGT begint op adres &B00**010**00000000000, decimaal 4096.

VDP(6)=&B111

De SGT begint op adres &B00**111**00000000000, decimaal 14336.

REGISTER 7

oproep actuele inhoud	PRINT PEEK(&HF3E6)
binair	11110100
toewijzing	VDP(6)=...

De betekenis van de bits is afhankelijk van de SCREEN-modus die u kiest.

REGISTER 7, SCREEN 0

De bits 0-3 bevatten informatie over de actuele achtergrondkleur en bit 4-7 over de voorgrondkleur. Bij SCREEN 1 zijn maar twee actuele kleuren mogelijk (voor- en achtergrondkleur), waarbij u kunt kiezen uit 15 beschikbare kleuren. U kunt deze kleurinformatie het eenvoudigst in hexadecimale vorm invoeren in register 7. Reken daartoe de 8 bits om in twee hexadecimale waarden van 4 bits elk. Zo is de informatie over de twee kleuren duidelijk herkenbaar en hoeft u maar 1 hexadecimaal getal te gebruiken. Bijvoorbeeld: VDP(7)=&HF1. De voorgrondkleur is &HF = decimaal 15 = wit; de achtergrondkleur is &H1 = decimaal 1 = zwart.

REGISTER 7, SCREEN 1, 2 of 3

In deze gevallen bepalen de bits 0-3 de kaderkleur en de bits 4-7 weer de voorgrondkleur. Ook hier is het het handigst het register te laden met een hexadecimaal getal. Net als bij SCREEN 0 is dat voldoende.

Vooropgesteld dat SCREEN 0 het actuele scherm is, is de kaderkleur wit (F=15 en en de achtergrondkleur zwart (1). Op SCREEN 1, 2 of 3 is de kaderkleur zwart en de kleur van de voorgrond wit.

REGISTER 8

oproep actuele inhoud	PRINT PEEK(&HF3E7)
binair	11001101
lezen	PRINT VDP(8)

We hebben al gezegd dat u geen waarden in register 8 kunt schrijven, u kunt het alleen lezen. Als er meer dan vier sprites op een horizontale lijn op het scherm staan, ontstaan er moeilijkheden. Nummer vijf en hoger zijn dan niet meer geheel te zien, waarbij de regel geldt dat de sprites met de laagste nummers voorrang hebben (die liggen in de ruimte gezien bovenop). Soms zijn zelfs bij acht sprites nog wel stukjes zichtbaar. Sprites met hogere nummers komen dan niet of slechts gedeeltelijk op het scherm. Zelfs bij 8 sprites kunnen nog delen van 'hogere' sprites zichtbaar zijn.

bit 0-4

Als het zover komt dat er meer dan 4 sprites op het scherm staan, dan bevatten deze bits het nummer van de vijfde sprite. Deze sprite voegt zich dan bij de sprites die al op het scherm staan; de storende overlapping is een feit (zie het programma in paragraaf 9.5).

bit 5

Bit 5 heeft de waarde 1 bij een botsing van twee sprites mits de machine-interrupt actief is (door of een 1 op bit 1 van register 5, of de BASIC-instructies SPRITE GOSUB en SPRITE ON). Deze bit is 0 als er geen botsing plaatsvindt. U moet wel uitkijken als u register 8 uitleest om te zien of er sprake is van een botsing: door de uitlezing wordt de interruptbesturing direct uitgeschakeld (vergelijkbaar met SPRITE OFF). Om dat te voorkomen, kunt u geheugenplaats &HF3E7 lezen want die komt steeds overeen met de inhoud van register 8.

bit 6

Deze bit is 1 als er meer dan vier sprites op een schermregel staan. Om zo snel mogelijk iets te kunnen ondernemen tegen dit vervelende effect moet u meteen uit de bits 0-4 het nummer van de storende sprite afleiden om tegenmaatregelen te kunnen nemen. Bit 6 is 0 als er geen sprake is van meer dan vier sprites op 1 beeldlijn. Let goed op het verschil tussen een schermregel en een beeldlijn. Een regel bestaat uit 8 lijnen, zoals een teken uit 8*8 punten bestaat.

bit 7

Bit 7 werkt indirect met alle VDP-instructies samen. Elke keer als een beeldschermoproep is beëindigd, krijgt deze bit de waarde 1 (50 keer per seconde). Als u in BASIC programmeert, is deze functie vanwege de traagheid van BASIC niet van belang. Pas in machinetaal kan deze bit belangrijke inlichtingen verstrekken.

Voorbeeld bij register 8

```
PRINT BIN$(VDP(8))    of in BASIC:

PRINT BIN$(PEEK(&HF3E7))

resultaat: 11111111
```

bit 0-4 (1111)

Er bevindt zich een sprite (nummer &B1111, decimaal 31) geheel of gedeeltelijk tussen vier andere sprites.

bit 5 (1)

Twee sprites zijn op elkaar gebotst (de bit is gezet).

bit 6 (1)

Bevestigt de melding van de bits 0-4: er bevinden zich meer dan vier sprites op een horizontale lijn.

bit 7 (1)

Het scherm wordt om de vijftigste seconde 'opgefrist' (refresh).

1.4 BASE

De programma's BASE 1-14 staan aan het einde van deze paragraaf.

Enige kennis van opbouw en gebruik van het video-RAM is wenselijk als we met deze instructie gaan werken. De instructie BASE is niet zelf verantwoordelijk voor tekst en grafiek, maar geeft precies aan hoe het VRAM is opgebouwd in de actuele SCREEN-modus. Bovendien kunt u met BASE het VRAM naar eigen inzicht inrichten. Eigenlijk is BASE dus meer een variabele dan een instructie. Al naar gelang de waarde die u toekent aan BASE wordt de structuur van het VRAM direct of indirect beïnvloed. BASE is een gedimensioneerd veld van 20 waarden. Meteen na inschakeling van de computer is dit veld al van getallen voorzien. Als we programma BASE 1 uitvoeren, is het resultaat:

```
 0 =     0     1 =     0
 2 =  2048     3 =     0
 4 =     0     5 =  6144
 6 =  8192     7 =     0
 8 =  6912     9 = 14336
10 =  6144    11 =  8192
12 =     0    13 =  6912
14 = 14336    15 =  2048
16 =     0    17 =     0
18 =  6912    19 = 14336
```

Op het eerste gezicht zegt deze rij getallen niet veel. Ze worden ook niet allemaal tegelijk gebruikt maar in groepjes van vijf. Die vier groepen komen overeen met de vier SCREEN-modi van de MSX.

0- 4	SCREEN 0
5- 9	SCREEN 1
10-14	SCREEN 2
15-19	SCREEN 3

De vijf verschillende waarden hebben elk betrekking op het beginadres van een bepaald geheugenbereik in het VRAM. De inhoud van elk bereik is:

0 indeling van het beeldscherm; het gaat steeds om 256 verschillende tekens

1 kleur van afgebeelde tekens; (niet in SCREEN 0 en 3)

2 vorm (pixel voor pixel) van de afzonderlijke tekens en grafische figuren

3 informatie over positie en kleur van sprites (niet in SCREEN 0)

4 vorm van sprites (niet in SCREEN 0)

De waarden 5-9 bevatten dezelfde informatie, maar dan voor SCREEN 1, 10-14 voor SCREEN 2 en 15-19 voor SCREEN 3. Uit het bovenstaande blijkt dat in SCREEN 0 en 3 slechts gedeelten van het VRAM worden benut met BASE. Daarom werken we verder in SCREEN 1. We geven nu van elk van de vijf waarden een voorbeeld.

1.4.1 Indeling van het beeldscherm

Het beginadres van de scherminhoud is volgens de BASE-tabel adres 6144. Om dat te controleren, wissen we het scherm en POKEn een 2 in adres 6144.

```
SCREEN 1

VPOKE 6144,2
```

Let er op dat u niet per ongeluk de instructie POKE gebruikt; we werken in het VRAM (video-RAM) en daar hoort de instructie VPOKE bij. Op het beeldscherm is niet veel veranderd, behalve dan dat er in de linkerbovenhoek een sprite staat in de vorm van een lachend gezichtje dat u nog wel kent van de smile-buttons. In de ASCII-tabel in bijlage 3 blijkt dat het om karakter 2 gaat. Als u niets ziet, kan het zijn dat de monitor niet goed is afgesteld. Probeer deze zo af te stellen dat ook de uiterste linker rand van het scherm zichtbaar wordt. We verschuiven de sprite naar het midden van de eerste regel. In SCREEN 1 passen 32 tekens op een regel. De middenpositie ligt dus 15 plaatsen verder dan het eerste teken. Voer in:

```
VPOKE 6144 + 15,2
```

Met deze sprite als uitganspunt zetten we vervolgens een hele kolom sprites op het scherm, van boven naar beneden. Voorbeeldprogramma BASE 2 regelt dat. Als u waarden in het VRAM POKEt, stoort de computer zich niet aan de functiebalk (de vijf functietoetsen) op regel 24. Staat die balk in een tekstscherm, dan heeft u maar 23 regels ter beschikking. Normaliter kunt u niet over deze regel heenschrijven. Nu verschijnt er ook een

lachebekje op regel 24. Deze moet u met VPOKE wissen, tenzij u de functiebalk opnieuw definieert. U kunt het scherm ook wissen met CLS of SCREEN 1. De sprite verdwijnt ook van de functiebalk als u op de SHIFT-toets drukt.

Zo kunnen we een tweede schermkader maken (het eigenlijke kader wordt bepaald door de derde parameter van de instructie COLOR). Een tweede kader kunnen we gebruiken als we het oorspronkelijke venster verkleinen met de instructie WIDTH en de functiebalk laten staan. Experimenteer maar eens met voorbeeld BASE 3. Links, rechts en onder is het venster omringd door sprites. Zolang u geen CLS of SCREEN gebruikt, blijft dat zo. U kunt de tweede rand weer wissen door de breedte van het scherm te veranderen of opnieuw waarden te POKEn in de betreffende beeldschermadressen. Uit het programma BASE 3 blijkt dat in SCREEN 1 het scherm loopt van adres 6144 (BASE(5)) tot adres 6911; dat zijn 768 geheugenplaatsen (24 regels * 32 tekens). Hoe zit het nu met SCREEN 1? Het begin van het beeldschermgeheugen vinden we met

```
PRINT BASE(0)

resultaat: 0
```

In SCREEN 0 is het beeldschermgeheugen 960 tekens lang (24 regels * 40 tekens). Dat zijn de adressen 0 tot 959. Tenslotte nog de grafiekschermen 2 en 3. Ook hier roept u met PRINT BASE de startadressen van het beeldschermgeheugen op: in SCREEN 2 wordt dat PRINT BASE(10) met als resultaat 6144 en in SCREEN 3 PRINT BASE(15) met als resultaat 2048. Het beeldschermgeheugen omvat in SCREEN 3 768 geheugenplaatsen. Nog een keer alles op een rijtje:

	begin	lengte	variabele
SCREEN 0	0	960	BASE(0)
SCREEN 1	6144	768	BASE(5)
SCREEN 2	6144	768	BASE(10)
SCREEN 3	2048	768	BASE(15)

1.4.2 Kleur van tekst en grafiek

BASE(1) heeft de waarde 0. Dat BASE(0) en BASE(1) beide de waarde 0 opleveren lijkt vreemd. De 0 van BASE(0) staat er alleen maar omdat in SCREEN 0 niet met verschillende kleuren kan worden gewerkt, behalve wanneer we de instructie COLOR gebruiken. Deze instructie is van toepassing op het hele scherm, net als bij SCREEN 1, en neemt maar 2 adressen in beslag. Daarom is er voor SCREEN 0 niet een apart kleurenpalet opgeslagen in het VRAM. We gaan nu verder met SCREEN 1. Voer de volgende regels in.

```
10   SCREEN 1
20   PRINT BASE(6)
```

resultaat: 8192

Het kleurenpalet begint op adres 8192 in het VRAM. Het bevat maar 32 tekens, die op z'n zachtst gezegd ongelukkig zijn verdeeld. Voer voorbeeld BASE 4 maar eens in. Al naar gelang het aantal en de vorm van de weergegeven tekens verandert het scherm in een warboel, die u misschien niet eens meer kunt ontcijferen. Om te kunnen zien waarom dat zo is, moeten we de oorspronkelijke toestand van de kleuren herstellen met SCREEN 1 (kleurverandering met COLOR helpt niet meer). Voeg aan het programma BASE 4 de regels van BASE 4a toe en start het met RUN. U ziet nu groepjes van acht tekens (bijvoorbeeld H tot 0 of 0 tot 7) die steeds een andere kleur krijgen. Zo hebben bijvoorbeeld de cijfers 6 en 7 dezelfde kleur, maar 8 heeft een andere kleur. 6 en 7 behoren tot hetzelfde cluster en bij 8 begint een nieuw. Hoe verloopt nu de invoer van de 32 kleurbytes? Stel dat u geheugenplaats 8198 wilt veranderen (daarin staat de kleurtoewijzing van de cijfers 0-7). U voert in:

```
VPOKE 8198,255
```

resultaat: witte ondergrond, wit teken

```
VPOKE 8198,0
```

resultaat: doorzichtige ondergrond, doorzichtig teken

```
VPOKE 8198,31
```

resultaat: witte ondergrond, zwart teken

in binaire vorm:

```
255 = 11111111
  0 = 00000000
 31 = 00011111
```

Door elk binair getal op te delen in twee groepen van 4 bits krijgen we twee nieuwe binaire getallen; bij de eerste waarde worden dat bijvoorbeeld de getallen 1111 en 1111, of decimaal 15 en 15. Het eerste getal is de code voor de voorgrondkleur (het teken zelf) en het tweede getal bepaalt de achtergrondkleur. Zowel voor- als achtergrondkleur zijn dus wit. In het derde voorbeeld zijn de twee kleuren 0001 en 1111, decimaal 1 en 15. Dat betekent een zwarte voorgrond en een witte achtergrond. Ter illustratie nog twee voorbeelden in omgekeerde volgorde. Daarna kunt u het zelf gaan proberen.

voorgrondkleur	13
achtergrondkleur	7
binair	(1101)(0111)
decimaal	215
instructie	VPOKE 8198,215

voorgrondkleur	7
achtergrondkleur	13
binair	(0111)(1101)
decimaal	125
instructie	VPOKE 8198,125

Vervolgens de kleurtoekenning in SCREEN 2. Uit de waarde van BASE(11) blijkt dat het kleurgeheugen voor grafiek met hoog oplossend vermogen begint op geheugenplaats 8192. In het kleurgeheugen zijn 6144 tekens opgeslagen (in SCREEN 2 kunnen ongeacht de grootte van de tekens 192 lijnen willekeurige kleuren krijgen; op 1 lijn passen 32 tekens; 192 * 32 = 6144). Voorbeeld BASE 5 demonstreert deze overvloed aan kleuren. Op het scherm ziet u een bont kleurenpalet; dit plaatje verduidelijkt de indeling van het VRAM. Eigenlijk kunt u met het blote oog niet alles zien want ieder gekleurd vakje van 8*1 pixels (een horizontaal balkje) bevat zowel een voor- als een achtergrondkleur. De opslag van die kleuren voor afzonderlijke tekens gaat precies zoals beschreven bij SCREEN 1: de eerste vier bits voor de voorgrondkleur en de laatste vier bits voor de achtergrondkleur. Daaruit kunnen weer hexadecimale waarden worden gedestilleerd om de invoer te vereenvoudigen. Een voorbeeld:

voorgrondkleur	7
achtergrondkleur	15
hexadecimaal	7 en F
binair	(0111)(1111)
decimaal	127

Tot slot nog de kleurtoewijzing in SCREEN 3. Dit 'grove' grafische scherm (minder oplossend vermogen dan SCREEN 2) bestaat uit 64*48 punten. In elke byte zijn twee kleurgegevens opgeslagen (voor- en achtergrondkleur). 1 Byte spreekt steeds twee pixels tegelijk aan zodat er 1536 (64/2*48) bytes/tekens nodig zijn om dit scherm op te bouwen. De kleurtoewijzing nog een keer in schema:

	begin	lengte	variabele
SCREEN 0	----	----	BASE(1)
SCREEN 1	8192	32	BASE(6)
SCREEN 2	8192	6144	BASE(11)
SCREEN 3	0	1536	BASE(16)

1.4.3 Pixelvorm van de afzonderlijke tekens

Omdat SCREEN 0 de tekstmodus is, hebben we hier alleen te maken met de vorm van de afzonderlijke tekens. De opslag daarvan begint op adres 2048 en neemt ookt 2048 geheugenplaatsen in beslag. Hoe wordt de vorm van een teken opgeslagen? Tekens worden door de computer behandeld als getallen. Een voorbeeld:

spatie	CHR$(32)
" (aanhalingsteken)	CHR$(34)
A	CHR$(65)

(zie s.v.p. ook hoofdstuk 4 over CHR$ en ASC). Programma BASE 6 demonstreert hoe de MSX letters en karakters opslaat. Resultaat:

```
32
80
136
136
248
136
136
0
```

Deze informatie is nog niet voldiende voor de weergave van een letterteken (in het voorbeeld een A) want de computer zet ieder decimaal getal om in een binair getal. Programma BASE 7 doet dat voor de letter A, met als resultaat:

```
00100000      1
01010000     1 1
10001000    1   1
10001000    1   1
11111000    11111
10001000    1   1
10001000    1   1
00000000
```

U vraagt zich misschien af waarom in eze 8*8 matrix de drie rechter kolommen leeg c.q. ongedefiniëerd zijn. Ten eerste blijft er een kolom vrij om te zorgen dat letters niet tegen elkaar aankomen. Dat de andere twee kolommen ook vrij blijven is een eigenschap van de hi-res grafiek van de MSX. Er moeten immers 40 tekenbreedtes op een beeldlijn passen. Met 40 tekens per beeldlijn en letters van 8 punten breed komen we uit op 320 pixels per regel, terwijl er maar 256 ter beschikking zijn. Met tekens van 6 pixels breed komen we wel uit (240). Voor de onderste lijn van de letter A geldt iets dergelijks. Lijn 7 is nodig voor de staartjes van de kleine letters zoals de j en de g. U kunt de

vorm van de letters binnen de 8*8 matrix naar eigen inzicht veranderen. Om dit karwei, dat zeer veel tijd kost, wat te vergemakkelijken hebben we in het boek een karaktergenerator opgenomen: het programma in paragraaf 9.14. In SCREEN 1 verloopt het proces op dezelfde manier als in SCREEN 0. De lengte van het geheugenblok is ook hier 2048 bytes, alleen begint het nu op 0. In deze schermmodus staan er in plaats van 40 maar 32 karakters op een regel. We kunnen de 8*8 matrix volledig benutten: 32*8=256 pixels. Dat is precies het oplossend vermogen van de MSX. Het ligt dus voor de hand voor SCREEN 1 een extra karakterset te maken. Die vindt u in bijlage 2. Als u wilt, kunt u deze karakters met behulp van de karaktergenerator intypen. Nu nog de grafiekschermen. We hebben in het voorgaande de vraag gesteld hoeveel geheugenplaatsen er nodig zijn om 256*192 pixels weer te geven. Tot nu toe was alleen het beginadres van elk geheugenblok bekend. Om de weergave van kleuren in SCREEN 2 te demonstreren, maken we gebruik van voorbeeld BASE 5. Hi-res pixelgrafiek wordt pas zichtbaar door gebruik te maken van kleuren. Dit voorbeeldprogramma moet dus worden aangepast zodat het tegelijkertijd kleur en hi-res grafiek weergeeft. Alleen op die manier wordt duidelijk dat er zowel voor- als achtergrondkleuren in het geding zijn. Voer programma BASE 8 in. Regel 20 zet puntsgewijs een zogenaamd bitpatroon op het scherm. Een pixel op het scherm komt overeen met een bit. Willen we de pixel zichtbaar maken, dan zetten we de bit (1). Het resultaat van BASE 7 is een voorbeeld van een bitpatroon. Daarna komen de kleuren er overheen. Per 8 punten op een horizontale lijn (1 byte) zijn maximaal twee kleuren toegestaan. Helaas is het niet mogelijk grafiek met hoog oplossend vermogen te gebruiken in SCREEN 3. BASE(17) is niet voorzien van waarden. De colour table (alle 16 kleuren) neemt SCREEN 3 al geheel in beslag. De grafiek met hoge oplossing nog eens in schema:

	begin	lengte	variabele
SCREEN 0	2048	2048	BASE(2)
SCREEN 1	0	2048	BASE(7)
SCREEN 2	0	6144	BASE(12)
SCREEN 3	----	----	BASE(17)

1.4.4 Positie en kleur van sprites

We komen nu toe aan de manier waarop sprites zijn opgeslagen in het VRAM. Gelukkig heeft de MSX een groot aantal instructies in BASIC waarmee we de vloeiend heen en weer bewegende sprites aanmerkelijk kunnen hanteren. De vierde BASE-waarde heeft betrekking op de SAT (sprite attribute table (toewijzingstabel)) in de vier schermmodi. Wat dit begrip inhoudt, wordt u al snel uit de voorbeelden duidelijk.

Met SCREEN 0 zijn we vlug klaar: hierin komen geen sprites voor dus is er ook geen toewijzingstabel. In SCREEN 1 begint de SAT op het VRAM-adres 6912 en is 128 bytes lang. De beschrijving en het gebruik van deze 128 bytes is in de andere SCREENs hetzelfde. In elke schermmodus kunnen maximaal 32 sprites op het scherm staan met de al genoemde beperking van maximaal 4 sprites per horizontale lijn. Als er een vijfde sprite op het scherm verschijnt, wist deze de sprite met het hoogste nummer. Als de nieuwe sprite over de oude heen is geschoven, staan de orspronkelijke vier sprites weer op het scherm alsof er niets is gebeurd. In de SAT staan vier gegevens voor elk van de 32 sprites (32*4=128 bytes).

teken	betekenis
1	Y-coördinaat
2	X-coördinaat
3	keuze van weer te geven sprite (zie de instructie SPRITE$ in paragraaf 5.8)
4	- kleur van de sprite - plaats van het aanstuurpunt van X,Y

Eerst iets meer over het vierde teken. Stel dat een sprite wit is (kleur 15)

decimaal 15 = binair 00001111

De bits 0-3 bevatten de kleurcode. De bits 4-6 hebben geen betekenis. Voor bit 7 moeten we nader ingaan op de verwerking van getallen door computers. Omdat de eerste twee tekens van de sprite-toekenning slaan op een van de 256*192 punten van het scherm lijkt het geen probleem een sprite punt voor punt over het scherm te laten glijden. Dat klopt voor een verticale beweging. Horizontaal ligt dat anders. De coördinaten van een sprite worden berekend vanuit de linker bovenhoek; dat klopt ook zolang een sprite van links naar rechts beweegt. Programma BASE 9 demonstreert dat. De sprite schuift rustig naar rechts het beeld uit en komt niet meer terug. Nu proberen we de omgekeerde beweging met voorbeeld BASE 10. De sprite komt keurig van rechts in beeld maar zodra positie 0,0 is bereikt, blijft hij staan wachten op verdere instructies. De zaak komt weer in beweging door bit 7 van het vierde teken de waarde 1 te geven (dat komt neer op plus 128); voor de computer een sein de patstelling op te heffen door de coördinaten anders te interpreteren.

x=x-32
y=y

De computer zet bij weergave van de sprites het nulpunt van de

coördinaten op -32 en daardoor verdwijnt de sprite meteen uit het gezichtsveld. Deze procedure verricht de MSX in BASIC automatisch als u negatieve waarden invoert maar daar is wel een grens aan. U kunt de FOR/NEXT-lus in regel 40 in het negatieve gebied laten doorlopen totdat op zeker moment de sprite aan de rechter beeldrand weer te voorschijn komt (zonder dat hij verder het scherm opkomt). U voorkomt dit onverwachte effect door het normale beeldformaat met niet meer dan 32 tekens te overschrijden (dus van -32 tot 287). Het is interessant te zien hoe ingewikkeld de computer bij deze procedure in het VRAM te werk moet gaan. Bekijk het resultaat van voorbeeld BASE 11. Het vervolg van het verhaal, waarin de sprite om de rechterhoek komt kijken, krijgen we te zien als we BASE 11a toevoegen.

1.4.5 De vorm van de sprites

Behalve de toekenning van kleur, positie en nummer moet natuurlijk ook de vorm van een sprite worden bepaald. Omdat in SCREEN 0 sprites niet voorkomen en in de andere schermmodi het geheugen op dezelfde manier wordt gebruikt, bespreken we alleen de geheugenopbouw in SCREEN 1.

Het geheugenbereik dat voor sprites is gereserveerd, begint op adres 14336 en beslaat 2048 bytes. Er kunnen 32 sprites in worden opgeslagen. De al eerder besproken beperking van vier sprites per regel blijft geldig. Als u tevreden bent met figurtjes die in een 8*8 matrix passen (in SCREEN ,0 of SCREEN ,1), dan kunt u in het VRAM 256 sprites definiëren (256*8=2048). In dat geval is sprite 0 opgeslagen op de adressen 14336-14343, sprite 1 op 14344-14351 en sprite 255 op 16377-16384. Voor de duidelijkheid kunt u de de spritevorm het beste binair (als bitpatroon) invoeren en dit vervolgens omzetten in decimale waarden om ruimte te besparen. Bijvoorbeeld:

binair	decimaal
00000000	0
01111110	126
01111110	126
01100110	102
01100110	102
01111110	126
01111110	126
00000000	0

Met de BASIC-instructie PUT SPRITE tovert u de kersverse sprite op het beeldscherm. Programma BASE 12 demonstreert dit. Het resultaat verschijnt midden op het scherm. Vergelijk het met het voorbeeld hierboven. Als we de omzetting van binair in decimaal

willen overslaan, kunnen we de waarden ook gelijk binair invoeren. In programma BASE 13 is dat gedaan. In deze paragraaf gaat het erom de opbouw van het geheugen te verkennen. Het is daarom nuttig instructies te gebruiken die het geheugen rechtstreeks aanspreken in plaats van de zeer uitgebreide set instructies (vooral de grafische) die ons in MSX-BASIC ter beschikking staat. Dat geldt ook de grafische instructie PUT SPRITE. Om een sprite op het scherm te krijgen moet u de vier toekenningswaarden invoeren (coördinaten, nummer en kleur). In dit geval gaat het om de eerste sprite (adres in de SAT: 6912). In de programma's BASE 12 en 13 veranderen we de regels 70 respectievelijk 90 in

```
70/90    VPOKE 6912,96
71/91    VPOKE 6913,128
72/92    VPOKE 6914,0
73/93    VPOKE 6915,15
```

Voor grote sprites kiest u een andere waarde voor de tweede parameter van de schermmodus; dus SCREEN ,2 of SCREEN ,3. Een sprite van 16*16 neemt vier maal zoveel punten in beslag als een sprite van 8*8. We zagen al eerder dat er 256 kleine sprites in het VRAM-bereik passen en 64 (256:4) grote sprites. Elke sprite bezet 16*2 adressen (8 punten per adres), dus alle sprites samen 2048 (64*16*2) adressen. Dat komt overeen met de voor sprites beschikbare geheugenruimte. De grote sprites worden in vier delen van 8*8 opgeslagen in het VRAM. De volgorde is linksboven, linksonder, rechtsboven en rechtsonder. We maken van ons vorige voorbeeld een sprite van 16*16 (vier keer hetzelfde bitpatroon). Daarnaast geven we gelijk de overeenkomstige decimale waarden.

sprite 1	00000000	00000000	sprite 3	0	0
	01111111	11111110		127	254
	01111111	11111110		127	254
	01100000	00000110		96	6
	01100000	00000110		96	6
	01100000	00000110		96	6
	01100011	11000110		99	198
	01100010	01000110		98	70
sprite 2	01100010	01000110	sprite 4	98	70
	01100011	11000110		99	198
	01100000	00000110		96	6
	01100000	00000110		96	6
	01100000	00000110		96	6
	01111111	11111110		127	254
	01111111	11111110		127	254
	00000000	00000000		0	0

Een mooi plaatje, maar wel arbeidsintensief. Nu hebben we nog een programma nodig om decimale getallen in te lezen. Daar hebben we BASE 14 voor. En met BASE 15 kunnen we 16*16 sprites rechtstreeks invoeren in het geheugen, zonder de omzetting binair-decimaal en zoner de grafische MSX-BASIC instructies. U ziet nu waarschijnlijk wel de waarde in van BASIC/instructies als PUT SPRITE en SPRITE$ en van een werkbesparende sprite-editor zoals besproken in dit hoofdstuk. Dat betekent niet dat u zelf niets meer hoeft te doen: enig inzicht in de opbouw van het VRAM is onontbeerlijk als u zelf wilt programmeren. Nog een alles op een rijtje:

SCREEN	SPRITES type	aantal	beginadr.	lengte	BASE-variabele
0	-	-	-	-	4
1,0/1,1	8*8	256	14336	2048	9
1,2/1,3	16*16	64	14336	2048	9
2,0/2,1	8*8	256	14336	2048	14
2,2/2,3	16*16	64	14336	2048	14
3,0/3,1	8*8	256	14336	2048	19
3,2/3,3	16*16	64	14336	2048	19

BASE 1

```
10 COLOR 1,15
20 SCREEN 0
30 FOR N=0 TO 19
40 PRINT N"=";BASE(N),
50 NEXT N
```

BASE 2

```
10 COLOR 1,15
20 SCREEN 1
30 FOR N=6144+15 TO 6144+751 STEP 32
40 VPOKE N,2
50 NEXT N
```

BASE 3

```
10 COLOR 1,15
20 SCREEN 1
30 KEY ON
40 WIDTH 20
50 COLOR 15,4,5
60 FOR N=6144 TO 6911
70 VPOKE N,2
80 PRINT STRING$(20,"A");
90 NEXT N
```

BASE 4

```
10 COLOR 1,15
20 SCREEN 1
30 FOR N=8192 TO 8192+32
```

```
         40 VPOKE N,INT(RND(1)*255)
         50 NEXT N

BASE 4a  21 FOR N=32 TO 255
         22 PRINT CHR$(N);
         23 NEXT N

BASE 5   10 COLOR  1,15
         20 SCREEN 2
         30 FOR N=8192 TO 8192+6144
         40 VPOKE N,INT(RND(1)*255)
         50 NEXT N

BASE 6   10 COLOR 1,15
         20 SCREEN 0
         30 FOR N=2048+(65*8) TO 2048+(65*8)+7
         40 PRINT VPEEK(N)
         50 NEXT N

BASE 7   10 COLOR 1,15
         20 SCREEN 0
         30 FOR N=2048+(65*8) TO 2048+(65*8)+7
         40 B$=BIN$(VPEEK(N))
         50 PRINT STRING$(8-LEN(B$),"0")+B$
         60 NEXT N

BASE 8   10 COLOR 1,15
         20 SCREEN 2
         30 FOR N=8192 TO 8192+6143
         40 VPOKE N,INT(RND(1)*255)
         50 VPOKE N-8192,INT(RND(1)*255)
         60 NEXT N

BASE 9   10 COLOR 1,15
         20 SCREEN 2,3
         30 SPRITE$(0)=STRING$(32,"U")
         40 FOR N=0 TO 255
         50 FOR M=0 TO 10
         60 NEXT M
         70 PUT SPRITE 0,(N,0)
         80 NEXT N
         90 GOTO 90

BASE 10  10 COLOR 1,15
         20 SCREEN 2,3
         30 SPRITE$(0)=STRING$(32,"U")
         40 FOR N=255 TO 0 STEP -1
         50 FOR M=0 TO 10
         60 NEXT M
```

```
70 PUT SPRITE 0,(N,0)
80 NEXT N
90 GOTO 90
```

BASE 11

```
10 COLOR 1,15
20 SCREEN 2,3
30 SPRITE$(0)=STRING$(32,"U")
40 FOR N=0 TO 255
50 VPOKE 6912,0
60 VPOKE 6913,N
70 VPOKE 6914,1
80 VPOKE 6915,1
90 NEXT N
100 GOTO 100
```

BASE 12

```
10 COLOR 1,15
20 SCREEN 1,0
30 FOR N=0 TO 7
40 READ A
50 VPOKE 14336+N,A
60 NEXT N
70 PUT SPRITE 0,(128,96)
80 GOTO 80
90 DATA 0,126,126,102,102,126,126,0
```

BASE 13

```
10 COLOR 1,15
20 SCREEN 1,0
30 FOR N=0 TO 7
40 READ A$
50 A$="&B"+A$
60 A=VAL(A$)
70 VPOKE 14336+N,A
80 NEXT N
90 PUT SPRITE 0,(128,96)
100 GOTO 100
110 DATA 00000000
120 DATA 01111110
130 DATA 01111110
140 DATA 01100110
150 DATA 01100110
160 DATA 01111110
170 DATA 01111110
180 DATA 00000000
```

BASE 14

```
10 COLOR 1,15
20 SCREEN 1,2
30 FOR N=0 TO 3
40 FOR M=0 TO 7
50 READ A
```

```
60 VPOKE 14336+(N*8) +M,A
70 NEXT M
80 NEXT N
90 VPOKE 6912,96
100 VPOKE 6913,128
110 VPOKE 6914,1
120 VPOKE 6915,1
130 GOTO 130
140 DATA 0,127,127,96,96,96,99,98
150 DATA 98,99,96,96,96,127,127,0
160 DATA 0,254,254,6,6,6,198,70
170 DATA 70,198,6,6,6,254,254,0
```

BASE 15

```
10 COLOR 1,15
20 SCREEN 1,2
30 FOR N=0 TO 15
40 READ A$
50 B$="&B"+LEFT$(A$,8)
60 VPOKE 14336+N,VAL(B$)
70 NEXT N
80 RESTORE
90 FOR N=0 TO 15
100 READ A$
110 B$="&B"+RIGHT$(A$,8)
120 VPOKE 14352+N,VAL(B$)
130 NEXT N
140 VPOKE 6912,96
150 VPOKE 6913,128
160 VPOKE 6914,1
170 VPOKE 6915,1
180 GOTO 180
190 DATA 0000000000000000
200 DATA 0111111111111110
210 DATA 0111111111111110
220 DATA 0110000000000110
230 DATA 0110000000000110
240 DATA 0110000000000110
250 DATA 0110001111000110
260 DATA 0110001001000110
270 DATA 0110001001000110
280 DATA 0110001111000110
290 DATA 0110000000000110
300 DATA 0110000000000110
310 DATA 0110000000000110
320 DATA 0111111111111110
330 DATA 0111111111111110
340 DATA 0000000000000000
```

2 VPEEK en VPOKE

2.1 VPEEK

De programma's VPEEK 1-7 staan aan het einde van deze paragraaf.

Het geheugen van de MSX bestaat uit drie delen. Het 32 Kbyte ROM met machinetaalroutines voert BASIC-instructies uit. Het lees- en schrijfgeheugen (RAM) bevat programma's, listings, actuele waarden van variabelen en parameters, de indeling van de functietoetsen, de actuele schermkleuren, etc. Het is bij de diverse merken verschillend van grootte. Bij de MSX maken we niet aleen onderscheid tussen ROM en RAM, maar ook tussen RAM en VRAM. Het VRAM (video-RAM) is een RAM-gedeelte dat speciaal voor grafiek is gereserveerd. Het is 16 Kbyte groot en bevat informatie over de vorm van letters en speciale tekens, de weergave van grafiek en tekst op het beeldscherm, de vorm en de actuele plaats van sprites, de pixelvorm van grafische figuren en de kleurtoekenning. Het VRAM heeft een eigen PEEK-instructie: VPEEK. Deze instructie werkt alleen in het VRAM, dus op de adressen 0-16383. Een VPEEK buiten dit bereik leidt tot de foutmelding 'Illegal function call'. In deze paragraaf zullen we ons niet bezighouden met de precieze indeling van het VRAM; dat is al gebeurd in het vorige hoofdstuk naar aanleiding van de variabele BASE. We demonstreren 1 keer hoe en waar in het VRAM de vorm van de karakters is opgeslagen in SCREEN 0 en hoe we deze met de instructie VPEEK kunnen oproepen. Programma VPEEK 1 resulteert in de uitvoer van een (enigszins abstract lijkend) rijtje getallen:

32, 80, 136, 136, 248, 136, 136, 0

Zo is de letter A opgeslagen in het VRAM. U bent deze getallen al tegengekomen in paragraaf 1.4.3. Een computer kan niets beginnen met decimale getallen; ze moeten eerst worden omgezet in nullen en enen. Het nadeel van deze eenzijdigheid wordt volledig opgeheven door de snelheid waarmee de computer informatie verwerkt (bij de MSX meer dan 10.000 schakelingen per seconde). De instructie BIN$ zet bovenstaande getallen om in binaire vorm (zie programma VPEEK 2). Met een beetje fantasie herkent u punt voor punt de letter A. Om dit duidelijker te zien kunt u de niet gevulde stukken van de uitgevoerde regels vullen met nullen (VPEEK 3). Helemaal duidelijk wordt het met VPEEK 4. Dezelfde procedure hebben we gebruikt om de hele karakterset van de computer op te roepen (zie het programma in paragraaf 9.12). Zo kunnen we een sprite of een heel scherm punt voor punt bekijken in het VRAM. In SCREEN 2 is dat niet zo simpel: de MSX is een kleurencomputer, waarbij grafiek en kleur

weliswaar samen op het beeldscherm komen, maar in het geheugen twee verschillende bereiken in beslag nemen. Experimenteer rustig verder met de programma's VPEEK 1-4 om sprites en karakters te bekijken. De indeling van het VRAM staat in paragraaf 1.4 (BASE). Tot slot nog een trucje: hoe kunnen we bij het lezen van het VRAM de invoer meerdere keren achter elkaar weergeven? Bekijk het voorbeeld VPEEK 5 maar eens. De letters kunnen ook op willekeurige plaatsen stuk voor stuk op het scherm worden gebracht met listing VPEEK 6. Zelfs de functiebalk op het scherm kunt u lezen en op een andere plaats weer laten verschijnen (listing VPEEK 7).

VPEEK 1

```
10 COLOR 1,15
20 SCREEN 0
30 FOR N=1 TO 8
40 PRINT VPEEK(2048+65*N-1);
50 NEXT N
```

VPEEK 2

```
10 COLOR 1,15
20 SCREEN 0
30 FOR N=1 TO 8
40 PRINT BIN$(VPEEK(2048+65*8+N-1))
50 NEXT N
```

VPEEK 3

```
10 COLOR 1,15
20 SCREEN 0
30 FOR N=1 TO 8
40 PRINT STRING$(8-LEN(BIN$(VPEEK(2048+65*8+N-1))),"0")
+BIN$(VPEEK(2048+65*8+N-1))
50 NEXT N
```

VPEEK 4

```
10 COLOR 1,15
20 SCREEN 0
30 FOR N=1 TO 8
40 A$=STRING$(8-LEN(BIN$(VPEEK(2048+65*8+N-1))),"0")
+BIN$(VPEEK(2048+65*8+N-1))
50 FOR M=1 TO 8
60 IF MID$(A$,M,1)="1" THEN PRINT CHR$(219);
ELSE PRINT " ";
70 NEXT M
80 PRINT
90 NEXT N
```

VPEEK 5

```
10 COLOR 1,15
20 SCREEN 0
30 WIDTH 40
40 A$="MSX-computer"
```

```
        50 PRINT A$
        60 FOR M=1 TO 11
        70 FOR N=1 TO LEN(A$)
        80 VPOKE M*20-1+N,VPEEK(N-1)
        90 NEXT N
        100 NEXT M
        110 LOCATE 0,10
```

VPEEK 6

```
10 COLOR 1,15
20 SCREEN 0
30 WIDTH 40
40 A$="MSX-computer"
50 PRINT A$
60 FOR M=1 TO 10
70 FOR N=1 TO LEN(A$)
80 VPOKE INT(RND(1)*940+20),VPEEK(N-1)
90 NEXT N
100 NEXT M
110 LOCATE 0,10
```

VPEEK 7

```
10 COLOR 1,15
20 SCREEN 0
30 WIDTH 40
40 FOR M=1 TO 10
50 FOR N=1 TO 40
60 VPOKE M*40+N,VPEEK(23*40+N-1)
70 NEXT N
80 NEXT M
```

2.2 VPOKE

De programma's VPOKE 1-7 staan aan het einde van deze paragraaf.

De instructie PEEK (gevolgd door het gewenste adres en voorafgegaan door PRINT) leest ROM of RAM uit. Schrijven in het geheugen is mogelijk met POKE; maar alleen in het RAM. Een POKE naar een ROM-adres heeft geen resultaat. We houden ons hier alleen bezig met alles in en om het beeldscherm en dus met het video-RAM. Logischerwijze is er de instructie VPOKE met betrekking tot het VRAM (video-RAM). De indeling van het VRAM kent u al uit paragraaf 1.4 over BASE. Twee korte voorbeelden om mee te beginnen. Eerst VPOKE 1. Dit programma is bedoeld om de instructie VPEEK te verduidelijken. Zoals u weet begint de opbouw van het scherm linksboven; in modus SCREEN 1 komt die plaats overeen met adres 0. Met VPEEK 1 zien we wat er op de eerste drie adressen in het VRAM staat.

```
PRINT VPEEK(0)      77 = M
PRINT VPEEK(1)      83 = S
PRINT VPEEK(2)      88 = X
```

De adressen 0-2 bevatten inderdaad de informatie die we eerst met PRINT "MSX" op de eerste schermregel hebben geschreven. Dat wordt duidelijk als we het scherm wissen en met programma VPOKE 2 de letters MSX weer in de eerste drie adressen van het VRAM POKEn. De getallen die voor de karakters staan, zijn ASCII-codes. Alle karakters zijn opgeslagen als ASCII-code; de spatie is bijvoorbeeld 32. Bekijk daarvoor ook de BASIC-instructies ASC en CHR$ (paragraaf 4.1 en 4.2). Om met VPOKE een letter op het beeldscherm te zetten, moeten we de betreffende ASCII-code invoeren. Met VPOKE 3 voeren we een zin in, die teken voor teken in de linker bovenhoek op het scherm verschijnt (in modus SCREEN 0). Dat lijkt onnodig ingewikkeld maar dat is het niet. Ten eerste gaan achter de simpele instructie PRINT "...." dezelfde gecompliceerde machinetaalstappen schuil als achter de instructies die we net hebben leren kennen. Ten tweede bespreken we VPOKE en willen we die instructie ook graag aan het werk zien. Ten derde is er in een tekstscherm geen BASIC-instructie om waarden te lezen (in tegenstelling tot de grafische schermen, waarin u de instructie POINT kunt gebruiken). Wilt u dat toch dan moet u met VPEEK werken. Voordat u nu een beetje gaat rondPOKEn, is een waarschuwing op zijn plaats. Zorg dat u de indeling van het VRAM kent! Zomaar een POKE kan verwoestende effecten hebben. Zie bijvoorbeeld programma VPOKE 4. Resultaat: veel letters beginnen een woekerachtige groei te vertonen en al gauw is er niet veel meer te lezen. Misschien is uw eerste gedachte de computer uit en weer aan te zetten. Eenvoudiger is met behulp van een SCREEN-instructie het scherm te wissen. Het VRAM bevindt zich dan weer in de uitgangstoestand en alles is in orde. U bent nu gewaarschuwd en kunt zelf veranderingen op het scherm aanbrengen met de instructie VPOKE in SCREEN 0. De bedoeling is alle spaties op het scherm te vervangen door rastertjes. De meer dan 1000 schermposities komen allemaal vol te staan met schaakbordjes (blokjes van 64 velden, of strikt genomen 48 velden omdat in SCREEN 0 elk karakter 6*8 pixels beslaat). Bekijk het resultaat van programma VPOKE 5. Hoe is dat mogelijk? In SCREEN 0 zijn de karakters in het geheugen opgeslagen vanaf adres 2048. Elk karakter beslaat 8 bytes. Het teken met code 32 (de spatie) neemt dus de adressen 2048+32*8+0 tot 2048+32*8+7 in beslag. Deze adressen krijgen om en om de waarden 85 en 170. Waarom juist deze getallen? Dat komen we aan de weet als we met BIN$ de binaire weergave van deze getallen bekijken.

```
BIN$(85) = 01010101
BIN$(170)= 10101010
```

Overal waar een 1 staat is nu in het spatieteken een pixel gezet. Bij een 0 blijft de betreffende pixel leeg. De eenvoudigste manier om de oorspronkelijke spatie terug te krijgen, is met een willekeurige SCREEN-instructie. Een tweede manier is om in de bewuste geheugenplaatsen weer een 0 te POKEn.

```
40 VPOKE 2048+32*8+N,0
50 VPOKE 2048+32*8+N+1,0
```

De instructie VPOKE is heel belangrijk als u in machinetaal werkt, want dan heeft u onder andere niet de beschikking over de BASIC-instructie PRINT. Bij weergave op het beeldscherm in BASIC is VPOKE niet zinvol. Dat blijkt uit programma VPOKE 6, dat de schrijfsnelheid van verschillende instructies in SCREEN 0 vergelijkt. Als het goed is, moet elke MSX hetzelfde resultaat opleveren:

SCREEN 0

PRINT	3.36 seconde
VPOKE	3.64 seconde
PRINT PRINT$	0.58 seconde
machinetaalroutines	0.02 seconde

Dat zijn aanzienlijke verschillen. In hoofdstuk 10 gaan we nader in op machinetaal en de grafische mogelijkheden daarvan. Bovenstaand experiment dient alleen ter illustratie van de traagheid van de instructie VPOKE.

Opmerkingen

- Het VRAM heeft een omvang van 16 Kbyte. Het gaat om de geheugenplaatsen 0 tot 16383. De waarde achter de instructie VPOKE mag u kiezen tussen 0 en 255. Valt een waarde buiten dit bereik, dan volgt de foutmelding Illegal function call.
- In de modus SCREEN 0 is niet het volledige VRAM in gebruik voor de opbouw van het schermbeeld. Met enige voorzichtigheid kunt dan ook het resterende gedeelte benutten om variabelen of meer schermgegevens op te slaan. In bijlage 1 (lijst van adressen) en paragraaf 1.4 staat welke delen van het geheugen beschikbaar zijn. Omdat een SCREEN-instructie onherroepelijk het hele VRAM wist, moet u daar heel voorzichtig mee zijn.
- Een fraai effect krijgt u door niet direct met BASIC aanspreekbare VRAM-delen als kader te gebruiken. Zo kunt u bijvoorbeeld de functietoetsen en de eventueel aanwezige marges (bij WIDTH kleiner dan de grootstse waarde) als kader gebruiken, zoals in voorbeeld VPOKE 7 is gebeurd. Gebruik in

aansluiting op dit programma in geen geval een van de volgende instructies: CLS, SCREEN n, WIDTH, KEY OFF of KEY ON. U bent dan meteen minstens een deel van het scherm kwijt.

- Ook als u vertrouwd bent met het VRAM kunt u nog op grote moeilijkheden stuiten. Een simpele handeling als het trekken van een simpele lijn op een grafiekscherm met VPOKE is al moeilijk genoeg. De MSX springt heel zuinig om met zijn geheugen (dat blijkt bijvoorbeeld bij de weergave van een teken: het bitpatroon (nullen en enen) wordt in de vorm van decimale waarden tussen 0 en 255 opgeslagen om ruimte te besparen). Verder is het lastig om de twee gescheiden delen van het VRAM voor grafiek en kleur op een zinvolle manier met elkaar te verbinden.

VPOKE 1

```
10 SCREEN 0
20 WIDTH 40
30 COLOR 1,15
40 PRINT "MSX"
50 PRINT
60 FOR N=0 TO 2
70 PRINT VPEEK(N);
80 NEXT N
```

VPOKE 2

```
10 SCREEN 0
20 WIDTH 40
30 COLOR 1,15
40 VPOKE 0,77
50 VPOKE 1,83
60 VPOKE 2,88
70 LOCATE 0,1
```

VPOKE 3

```
10 SCREEN 0
20 WIDTH 40
30 COLOR 1,15
40 PRINT
50 PRINT
60 INPUT "voer een korte zin in:           ";A$
70 PRINT
80 PRINT "nu wordt de zin in het VRAM vanaf
 geheugenplaats 0 gePOKEt"
90 FOR N=1 TO 1000
100 NEXT N
110 FOR N=1 TO LEN(A$)
120 VPOKE N-1,ASC(MID$(A$,N,1))
130 NEXT N
```

VPOKE 4

```
10 SCREEN 0
```

```
20 WIDTH 40
30 COLOR 1,15
40 FOR N=1 TO 1000
50 VPOKE 2048+INT(RND(1)*2048),INT(RND(1)*255)
60 NEXT N
70 LIST
```

VPOKE 5

```
10 SCREEN 0
20 COLOR 1,15
30 FOR N=0 TO 7 STEP 2
40 VPOKE 2048+32*8+N,170
50 VPOKE 2048+32*8+N+1,85
60 NEXT N
```

VPOKE 6

```
10 CLEAR 200,50000!
20 KEY OFF
30 SCREEN 0
40 WIDTH 40
50 COLOR 1,15
60 TIME=0
70 FOR N=1 TO 960
80 PRINT "*";
90 NEXT N
100 A=TIME
110 CLS
120 TIME=0
130 FOR N=0 TO 959
140 VPOKE N,42
150 NEXT N
160 B=TIME
170 CLS
180 TIME=0
190 FOR N=1 TO 24
200 PRINT STRING$(40,"*");
210 NEXT N
220 C=TIME
230 CLS
240 FOR N=1 TO 12
250 READ A$
260 POKE 49999!+N,VAL("&H"+A$)
270 NEXT N
280 DEFUSR=50000!
290 TIME=0
300 D=USR(0)
310 D=TIME
320 DATA 3E,2A,21,00,00,01,C0,03,CD,56,00,C9
330 CLS
340 PRINT"het duurde in seconden:"
350 PRINT
```

```
360 PRINT"met PRINT          :";:PRINT USING "##.##";A/50
370 PRINT"met VPOKE          :";:PRINT USING "##.##";B/50
380 PRINT"met PRINT STRING$:";:PRINT USING "##.##";C/50
390 PRINT"met MACHINETAAL   :";:PRINT USING "##.##";D/50
```

VPOKE 7

```
10 SCREEN 0
20 KEY ON
30 COLOR 1,15
40 WIDTH 34
50 FOR N=0 TO 976
60 VPOKE N,ASC("*")
70 NEXT N
80 LOCATE 0,23
90 FOR N=1 TO 24
100 PRINT
110 NEXT N
```

3 MSX-BASIC en de twee tekstschermen

3.1 CLS

Het programma CLS 1 staat aan het einde van deze paragraaf en COLOR 4 aan het einde van paragraaf 3.4.

Met de instructie CLS (clear screen) kunt u in alle SCREEN-modi het hele scherm wissen. Wel blijft de functiebalk zichtbaar, mits deze met KEY ON is ingeschakeld. Ook de cursor blijft normaal op het scherm tenzij hij al was uitgeschakeld. In dat geval komt hij eventueel terug met LOCATE ,,1. Ook de instructie SCREEN wist het scherm, maar daar blijft het niet bij. Het hele VRAM komt weer in de uitgangstoestand zodat alle zaken die u veranderd had (bijvoorbeeld de karakterset) verloren gaan. Voorzichtigheid is dus geboden. Een primitief middel om het scherm te wissen is het te vullen met lege regels door middel van PRINT-instructies. Met CLS gaat het veel sneller. Ook bij kleurtoewijzing met COLOR hebben we CLS nodig. In SCREEN 0 en 1 heeft de instructie COLOR meteen effect maar in de grafische modi ligt dat anders. Een verandering van de tweede parameter (achtergrondkleur) van COLOR is dan niet voldoende. Met CLS kunnen we eerst de achtergrondkleur veranderen (zie programma CLS 1). Hierbij nog twee opmerkingen.

- Na CLS wordt de laatst aangegeven achtergrondkleur (de tweede parameter van de instructie COLOR) de schermkleur. Dit geldt in beide grafische modi.
- Het lukt niet om met de toetsen SHIFT en HOME (tegelijk indrukken) dit effect in de directe modus te bereiken en ook niet met PRINT CHR$(12) in de programmamodus (zie programma COLOR 4).

Elke keer als SCREEN verandert, wordt het scherm gewist. U kunt dit omzeilen door in plaats van SCREEN direct de betreffende routines in het ROM aan te roepen.

SCREEN 0	&H0078
SCREEN 1	&H007B
SCREEN 2	&H007E
SCREEN 3	&H0081

Als dit u niet direct duidelijk is, voer dan in:

```
SCREEN 0
DEFUSR=&H007E
A=USR(0)
```

Als u nu tekst intypt, komt die als grafiek op het scherm. Met CLS kunt u dit weer wissen. Met SCREEN 0 komt u weer in de tekstmodus, maar dan is het scherm wel meteen leeg. Om dat laatste te voorkomen moet u de machinetaalroutine voor SCREEN 0 gebruiken en in de directe modus de regels

```
DEFUSR=&H0078
A=USR(0)
```

invoeren. Daarmee bent u weliswaar weer bij het uitgangspunt terug, maar u kunt nu tenminste lezen wat u zonet blind heeft ingetypt.

CLS 1

```
10 SCREEN 2
20 COLOR 15,1
30 DRAW "bm 0,0"
40 OPEN "grp:" FOR OUTPUT AS #1
50 PRINT #1,"   alleen verandering van de
              voorgrondkleur"
60 FOR N=1 TO 1000
70 NEXT N
80 CLS
90 PRINT #1,"   nu ook, met behulp van CLS,
           verandering van de
                achtergrondkleur"
100 FOR N=1 TO 1000
110 NEXT N
120 COLOR 1,4
```

3.2 KEY ON/OFF

De programma's KEY ON 1-3 staan aan het einde van deze paragraaf.

Na inschakeling gaat de MSX automatisch naar SCREEN 0, een van de twee tekstschermen. Er passen 37 tekens op een regel (met WIDTH uitbreidbaar tot 40). 23 van de 24 schermregels kunt u rechtstreeks beschrijven. Als u toe bent aan de 24e regel (de functiebalk) scrollt het scherm automatisch een regel verder want de regel met de functietoetsen is een vast onderdeel van het het scherm en er kan niet overheen worden geschreven. Met de MSX-instructie KEY OFF komt regel 24 wel vrij. Met KEY ON werkt u weer op 23 regels. Normaal ziet u op de functiebalk de toetsen 1-5. Om de functietoetsen 6-10 te zien drukt u gelijktijdig een functietoets en een van de SHIFT-toetsen in. Bij de grootste

regellengte krijgt u van elke functie de eerste 7 tekens te zien. Daarvan blijven er nog maar 3 over bij WIDTH 20. De functies zijn dan nauwelijks meer te herkennen, maar daarvoor heeft de MSX dé instructie KEY LIST.

SCREEN	WIDTH	aantal tekens per functietoets
0	40	7
0	37	6
0	20	3
1	32	5
1	29	5

Op de grafische schermen staat geen functiebalk. Om deze toch op regel 24 te krijgen moeten we het functietoetsgeheugen op deze regel uitlezen. De inhoud van de 10 functietoetsen is opgeslagen op de adressen &HF87F-&HF91E. Omdat er 32 tekens op 1 regel passen en er 5 functietoetsen tegelijk op het scherm komen, krijgt elke functie 5 tekens.

functietoets	adres
1	&HF87F - &HF88E
2	&HF88F - &HF89E
3	&HF89F - &HF8AE
4	&HF8AF - &HF8BE
5	&HF8BF - &HF8CE
6	&HF8CF - &HF8DE
7	&HF8DF - &HF8EE
8	&HF8EF - &HF8FE
9	&HF8FF - &HF90E
10	&HF90F - &HF91E

Tegelijk met de invoer van het functietoetsgeheugen moet de status van de SHIFT-toets worden gecontroleerd, zodat of de toetsen 1-5 of de toetsen 6-10 op regel 24 komen te staan. Hierbij is een nieuwe instructie gebruikt, INP. Die bespreken we niet, daar is andere literatuur voor. Hier alleen een voorbeeld van de manier waarop op een grafisch scherm de SHIFT-toets wordt uitgelezen met INP (zie programma KEY ON 1). Tot slot enige opmerkingen.

- Net als in programma KEY ON 1 al gebeurde, kunt u altijd de standaard inhoud van de functietoetsen uitlezen uit de adressen &HF87F tot &HF91E (zie KEY ON 2).
- De standaardbetekenis van de functietoetsen is in het ROM opgeslagen op de adressen &H13A9-&H1448. Als u die wilt inschakelen, voer dan de volgende twee instructies in (direct of in een programma):

```
DEFUSR=&H003E
A=USR(0)
```

gevolgd door SHIFT.

- Als u werkt in SCREEN 0 of 1 en de weergave van de functiebalk is ingeschakeld (begintoestand of KEY ON) dan kunt u regel 24 alleen veranderen met VPOKE, zoals in programma KEY ON 3 wordt getoond. De begintoestand (met normale weergave van de functiebalk) kunt u op vier manieren herstellen:

instructies	CLS SCREEN KEY ON
toetsen	SHIFT en HOME tegelijk

- Ook tijdens de uitvoering van een programma verschijnen de functietoetsen 6-10 als u SHIFT indrukt.

KEY ON 1

```
10 COLOR 1,15
20 SCREEN 2
30 OPEN "grp:" FOR OUTPUT AS #1
40 DRAW "bm 12,96"
50 PRINT #1,"funktietoetsen op SCREEN 2"
60 PRINT #1,"  verandering met behulp van
              SHIFT"
70 Z=0
80 FOR N=10 TO 210 STEP 50
90 DRAW "bm =n;,183"
100 Z=Z+1
110 A$=""
120 FOR M=1 TO 5
130 A$=A$+CHR$(PEEK(Z*16+&HF86F+M-1+(ZZ*80)))
140 NEXT M
150 PRINT #1,A$;
160 NEXT N
170 A=INP(&HAA) AND &HF0
180 OUT &HAA,A OR 6
190 B=INP(&HA9)
200 IF B AND 1 THEN IF ZZ=1 THEN ZZ=0:
LINE(10,183)-(255,191),15,BF
210 IF NOT B AND 1 THEN IF ZZ=0 THEN ZZ=1:
LINE(10,183)-(255,191),15,BF
220 GOTO 70
```

KEY ON 2

```
10 SCREEN 0
20 COLOR 1,15
30 A$=STRING$(16,32)
40 FOR N=5033 TO 5192 STEP 16
50 FOR M=1 TO 16
60 IF PEEK(N-1+M)>31 THEN MID$(A$,M,1)
=CHR$(PEEK(N-1+M)) ELSE MID$(A$,M,1)=" "
70 NEXT M
80 PRINT A$
90 NEXT N
```

KEY ON 3

```
10 KEY ON
20 SCREEN 0
30 COLOR 1,15
40 FOR N=920 TO 959
50 VPOKE N,INT(RND(1)*255)
60 NEXT N
```

3.3 WIDTH

Het programma WIDTH 1 staat aan het eind van deze paragraaf.

Als u de MSX hebt aangezet en het MSX-logo is verdwenen, bevindt u zich in modus SCREEN 0. De schermbreedte bedraagt 37 tekens en in regel 24 staat de functiebalk. Bij de meeste computers moet u stevig in de weer met POKE-instructies om het aantal tekens per regel (N) te veranderen. De MSX heeft daarvoor een extra BASIC-instructie:

```
WIDTH N
```

N kan de volgende waarde krijgen.

SCREEN 0	WIDTH 1 tot 40
SCREEN 1	WIDTH 1 tot 32
SCREEN 2/3	WIDTH is niet van toepassing/ WIDTH slaat automatisch op SCREEN 0 of 1

Als u dit bereik overschrijdt, volgt de foutmelding Illegal function call. Het verschil tussen SCREEN 0 en SCREEN 1 is het aantal punten per lettermatrix. Dit is al besproken in paragraaf 1.4.3. Om bij 40 tekens per regel geen stukjes kwijt te raken hebben de karakters een breedte van 6 punten. Als u zelf een nieuwe karakterset ontwerpt in een 8*8 matrix moet u niet raar opkijken als karakters in SCREEN 0 onvolledig op het scherm verschijnen. Met de standaard karakterset van 6*8 punten lijkt in SCREEN 1 een regel wel een beetje uitgerekt omdat voor elk teken 8*8 punten beschikbaar zijn. De karaktergenerator (het programma in paragraaf 9.14) werkt dan ook met SCREEN 1. Als u de computer aanzet en meteen overgaat op SCREEN 1, zult u na enig telwerk vaststellen dat ook in deze modus niet automatisch het maximum aantal tekens per regel (32) wordt gebruikt maar slechts 29 (WIDTH 29). De oorzaak hiervan is het verschil tussen de in diverse landen heersende normen voor de opbouw van t.v.-beelden (PAL, NTSC, SECAM). Wij hebben te maken met het PAL-systeem en de internationale MSX-norm sluit daar niet helemaal goed op aan. Cirkels worden ellipsen en bij het maximale aantal karakters per regel kunnen er een paar achter de schermrand verdwijnen. Om met die internationale verschillen rekening te houden, gebruikt elke MSX bij inschakeling maar een gedeelte van de beschikbare regelbreedte. Kijkt u zelf wat er gebeurt als u met een grotere regelbreedte wilt werken (40 tekens in SCREEN 0 of 32 in SCREEN 1). Bij een monitor zijn beeldhoogte en beeldbreedte instelbaar (vertical width en horizontal width). In principe kan dan het beeld zo worden aangepast dat alle karakters in beeld zijn en dat cirkels echt rond zijn. Het 6*8 karakter van SCREEN 0 en het 8*8

karakter van SCREEN 1 wordt nog eens geïllustreerd met voorbeeld WIDTH 1. De instructie WIDTH heeft uiteraard ook invloed op regel 24. Als functietoets 1 (=COLOR 15,4,4) zijn standaardbetekenis heeft, is na inschakeling (SCREEN 0, WIDTH 37) het hele woord COLOR zichtbaar, maar na de instructie WIDTH 10 staat alleen de C er nog. In dit geval is het verstandiger met SCREEN ,,0 de functiebalk weg te halen en daardoor zelfs een extra regel ter beschikking te krijgen. U kunt de regelbreedte ook instellen door de gewenste waarde te POKEn in een van de volgende adressen:

&HF3AE	SCREEN 0
&HF3AF	SCREEN 1
&HF3B0	actuele scherm

Een POKE op &HF3AE of &HF3AF heeft pas effect als u het betreffende SCREEN opnieuw heeft opgeroepen. Met PRINT PEEK kunt u de actuele schermbreedte opvragen. Met de instructies BASE en VDP kunt u verschillende schermen (in het voorbeeld 13) achter elkaar uit het VRAM oproepen en bekijken (zie het programma in paragraaf 9.2). U kunt deze schermen alleen bekijken en niet veranderen met WIDTH. Daarvoor zou u zo'n scherm direct moeten aanspreken met POKE. De ingevoerde waarden voor de schermbreedte blijven van kracht zolang ze niet worden gewijzigd, ook als u tussentijds overschakelt naar een ander SCREEN. Voor de volledigheid vermelden we nog dat SCREEN 2 een onveranderlijke regelbreedte van 32 tekens heeft. Op dit grafische scherm zou u natuurlijk wel door afzonderlijke pixels aan te spreken de karakters anders naast elkaar kunnen zetten. In SCREEN 3 is dat niet interessant omdat daarin maar 8 tekens op een regel passen (het meest geschikt voor titels).

```
WIDTH 1   10 COLOR 1,15
          20 SCREEN 0
          30 PRINT "    definitie van een vierkant"
          40 FOR N=1 TO 8
          50 READ A$
          60 VPOKE 2048+254*8+N-1,VAL("&B"+A$)
          70 NEXT N
          80 LOCATE 0,10
          90 PRINT "    op SCREEN 0 als volgt"
          100 PRINT
          110 PRINT"                  "; CHR$(254)
          120 FOR N=1 TO 1000
          130 NEXT N
          140 SCREEN 1
          150 PRINT "definitie van een vierkant"
          160 RESTORE
          170 FOR N=1 TO 8
          180 READ A$
          190 VPOKE 254*8+N-1,VAL("&B"+A$)
          200 NEXT N
          210 LOCATE 0,10
          220 PRINT "en zo op SCREEN 1"
          230 PRINT
          240 PRINT "          ";CHR$(254)
          250 GOTO 250
          260 DATA 11111111
          270 DATA 10000001
          280 DATA 10000001
          290 DATA 10000001
          300 DATA 10000001
          310 DATA 10000001
          320 DATA 10000001
          330 DATA 11111111
```

3.4 COLOR

De programma's COLOR 1-4 staan aan het einde van deze paragraaf.

Elke MSX beschikt over 15 kleuren, te gebruiken als voorgrond-, achtergrond- en kaderkleur.

0	doorzichtig
1	zwart
2	groen
3	lichtgroen
4	donkerblauw
5	lichtblauw
6	donkerrood
7	cyaan
8	rood
9	lichtrood
10	donkergeel
11	lichtgeel
12	donkergroen
13	magenta
14	grijs
15	wit

'Doorzichtig' betekent dat de betreffende plaats geen kleur heeft. U ziet dan de kleur van de volgende van de drie kleurparameters (COLOR voor,achter,kader). U kunt zich deze kleuren het beste voorstellen als drie lagen, waarbij de kaderkleur de achterste laag is (het 'diepst' in het scherm) en de voorgrondkleur (de tekens) de voorste laag. Als u een van deze lagen doorzichtig maakt, betekent dat dat de kleur van de volgende laag zichtbaar wordt. Kleur 0 voor de kader maakt deze kleurloos (u ziet dan alleen de kleur van het scherm). Experimenteer in SCREEN 1, 2 of 3 met een 0 voor de verschillende parameters om het transparante effect te ontdekken. Bij COLOR 0 krijgt het hele scherm de standaard kaderkleur (4). COLOR 0,0,0 wil niet zeggen dat het beeldscherm helemaal doorzichtig wordt. Er wordt dan helemaal geen kleur toegekend, zodat bij een juiste schermafstelling de kleur van het scherm zelf overblijft.

Bijna iedere kleur is in een lichte en een donkere tint voorhanden. Daarmee kunt u tamelijk eenvoudig een ruimtelijk effect bereiken, zoals het doosje in voorbeeld COLOR 1 laat zien. Het is wel van belang om te zorgen dat er niet meer dan twee kleuren op een veldje van 8*1 komen te staan, omdat de kleuren anders snel door elkaar gaan lopen wat een afgrijselijk resultaat oplevert (zie programma CIRCLE 4). De instructie COLOR heeft drie parameters.

```
COLOR A,B,C
```

- A voorgrondkleur (schrijfkleur)
- B achtergrondkleur
- C kaderkleur (als we niet in SCREEN 0 werken, zorgt deze parameter voor een kadertje boven- en onderaan het scherm (tenzij deze kleur gelijk is aan de achtergrondkleur)

Als u de MSX aanzet, is de standaard kleureninstelling COLOR 15,4,4: witte letters op een donkerblauwe achtergrond met een donkerblauw kader. De laatste is niet zichtbaar in SCREEN 0. U kunt deze instelling altijd oproepen met F6 (= functietoets 1 + SHIFT). Het voordeel van deze kleuren is dat de combinatie wit/blauw op een zwart-wit t.v. of een monochrome monitor voor voldoende contrast zorgt. Om de kaderkleur te veranderen en de andere kleuren ongemoeid te laten is het voldoende komma's te gebruiken.

```
10 SCREEN 1
20 COLOR ,,15
```

Stel dat voor- en achtergrondkleur gelijk zijn. U schrijft bijvoorbeeld met COLOR 15,15 wit op wit zodat er niets van te zien is. Al uw pogingen om iets op het scherm te krijgen blijven vruchteloos tot u met F6 de standaardkleuren oproept. U ziet dan ook meteen wat u hebt ingetypt om het probleem op te lossen. COLOR 15,15 betekent niet dat het scherm wordt gewist maar dat voor- en achtergrondkleur dezelfde waarde aannemen (in dit geval wit). We kunnen gebruik maken van dit effect door een beeld onzichtbaar op te bouwen en het vervolgens in 1 klap te voorschijn te laten komen. Programma COLOR 2 demonstreert dit effect. We bespreken nu de mogelijkheden van COLOR in de afzonderlijke schermmodi.

SCREEN 0

SCREEN 0 kent alleen een voor- en een achtergrondkleur. Dat er maar twee kleuren mogelijk zijn, is al besproken bij de instructie BASE. Daar zagen we dat SCREEN 0 geen bijbehorende kleurentabel in het VRAM heeft. De instructie COLOR werkt direct, zonder voorafgaande andere instructie (bijvoorbeeld CLS). Na omschakeling op een andere kleur veranderen niet alleen nieuw ingevoerde tekens maar ook de tekens die al op het scherm stonden.

SCREEN 1

Bij SCREEN 1 hoort een beeldkader zodat u alle drie de parameters kunt gebruiken. Ook hier heeft de instructie COLOR direct

resultaat; eerst CLS is niet nodig. Met een doorzichtige achtergrond (kleur 0) komt de kaderkleur naar voren als achtergrond. De instructie COLOR heeft geen invloed op de clusters van acht tekens waarvan de kleur al tevoren is gedefinieerd. Die informatie is immers opgeslagen in de colour table van het VRAM. Alleen tekens die niet op die manier zijn opgeslagen, veranderen mee van kleur.

SCREEN 2 en 3

Als een programma aan het werk is, kunt u alleen de kaderkleur veranderen. Een verandering van de voorgrondkleur wordt pas zichtbaar na een nieuwe invoer. In tegenstelling tot de tekstschermen behouden eerder ingevoerde tekens hun kleur na verandering van de eerste kleurparameter. Alle grafische instructies krijgen direct de actuele voorgrondkleur (behalve PRESET: achtergrondkleur!), tenzij zo'n instructie een eigen kleur heeft gekregen. Programma COLOR 3 geeft daarvan een voorbeeld. In de regels 30 en 50 werken de grafische instructies met de kleur die in regel 10 is aangegeven door COLOR (kleur 1). De instructie in regel 40 wordt uitgevoerd met een eigen kleur (15 = wit). Die kleur geldt alleen voor deze ene instructie. De Met COLOR ingestelde kleuren blijven gelden tot nader (COLOR-)order. Als we parameter 2 (achtergrondkleur) wijzigen, is het resultaat nog duidelijker. De kleurverandering wordt pas uitgevoerd op het scherm als dit is gewist met SCREEN ... of CLS (houd er rekening mee dat deze instructies verschillende gevolgen hebben, zoals we al zagen in paragraaf 1.1 en 3.1).

Nog twee belangrijke tips.

- Als u in machinetaal programmeert en u kunt of wilt de instructie COLOR niet gebruiken, dan moet u de gewenste waarden POKEn in de adressen

&HF3E9	voorgrondkleur
&HF3EA	achtergrondkleur
&HFEEB	kaderkleur

  De combinaties voorgrond-achtergrond (in SCREEN 0) en voorgrond-kader (in SCREEN 1 tot 3) kunt u ook rechtstreeks invoeren in register 7 van de video display processor. Wilt u in de hiervoor genoemde volgorde de kleuren wit (15) en zwart (1) gebruiken, dan kan dat eenvoudig met

  ```
  VDP(7)=&H1F      (let op de volgorde)
  ```

- De CLS/HOME-toets en de instructie PRINT CHR$(12) hebben op de grafische schermen niet dezelfde betekenis als de MSX-BASIC-instructie CLS, die u gebruikt om het scherm te wissen, en daarna eventueel de achtergrondkleur van een grafisch scherm te veranderen. Deze toets en instructie werken dus alleen in SCREEN 0 en 1 (zie programma COLOR 4).

COLOR 1

```
10 COLOR 15,1
20 SCREEN 2
30 IF Z=1 THEN C=1 ELSE C=3
40 DRAW "bm 112,96 c=c; r32 u32 l32 d32"
50 PAINT (140,86),3
60 IF Z=1 THEN C=1 ELSE C=12
70 DRAW "bm 144,96 c=c; e32 u32 n g32 l32 g32 r32 d32"
80 PAINT(154,76),12
90 PAINT(120,60),12
100 IF Z=1 THEN C=1 ELSE C=2
110 DRAW "bm 112,96 c=c; r32 g16 l32 e16"
120 PAINT(116,103),2
130 IF Z=1 THEN GOTO 150 ELSE Z=1
140 GOTO 30
150 GOTO 150
```

COLOR 2

```
10 KEY OFF
20 COLOR 15,15
30 SCREEN 0
40 WIDTH 40
50 FOR N=1 TO 220
60 PRINT "MSX ";
70 NEXT N
80 COLOR 1,15
```

COLOR 3

```
10 COLOR 1,4
20 SCREEN 2
30 CIRCLE(128,96),20
40 CIRCLE(128,96),40,15
50 CIRCLE(128,96),60
60 GOTO 60
```

COLOR 4

```
10 COLOR 1,4
20 SCREEN 2
30 CIRCLE(128,96),20
40 PRINT CHR$(12)
50 GOTO 50
```

3.5 TAB

De programma's TAB 1-3 staan aan het einde van deze paragraaf.

Elke schrijfmachine heeft een tabulatortoets. Deze verschuift de wagen over een tevoren ingestelde afstand. Als de tabulatorafstand ook nog instelbaar is, kunt u mooie tabellen maken. De MSX beschikt over zo'n faciliteit. De toets met de letters TAB (of een dubbele pijl naar rechts) laat de cursor 8 plaatsen naar rechts opschuiven. Daarnaast is er in MSX-BASIC ook nog een programmeerbare tabulator die u in een PRINT-instructie met de letters TAB kunt oproepen. De instructie heeft dan als parameter het aantal spaties na de laatste cursorpositie.

```
PRINT TAB(10);"MSX"
PRINT TAB(20);"MSX"
```

Het is ingewikkelder als het startpunt voor de printuitvoer niet samenvalt met het begin van de regel. De functie TAB werkt altijd vanaf het actuele regelbegin (=TAB(0)). Voorbeelden:

```
10 CLS
20 PRINT TAB(10);"MSX"
```

de uitvoer begint op regel 1, cursorpositie 10+1

```
10 CLS
20 LOCATE 10,0
30 PRINT TAB(15);"MSX"
```

de uitvoer begint op regel 1, cursorpositie 15+1

```
10 CLS
20 LOCATE 10,0
30 PRINT TAB(5);"MSX"
```

de uitvoer begint op regel 1, cursorpositie 10+1 (!). TAB(5) heeft een kleinere waarde dan het voorafgaande LOCATE 10,0 en wordt daarom genegeerd.

In deze voorbeelden hebben we bij de cursorposities steeds 1 opgeteld omdat de computer bij 0 begint te tellen. Cursorpositie 1 is dus eigenlijk schermpositie 0. Het bereik van de TAB-parameter is 0-255. U kunt de cursor snel vooruit bewegen, zelfs over meerdere regels.

```
PRINT TAB(0);"MSX"
```

In deze regel voegt TAB(0) niets toe, dus de uitvoer start gewoon

op het begin van de regel. Als u de regels 40 tekens breed maakt (met WIDTH 40) heeft

```
PRINT TAB(40);"MSX"
```

als resultaat dat de letters MSX op de tweede regel verschijnen:

volgende regel	TAB(0) t/m TAB(39)
daarop volgende regel	TAB(40) t/m TAB(79)

Als u buiten het bereik van de TAB-parameter komt, volgt de foutmelding Illegal function call.

Het lijkt heel eenvoudig om met TAB gegevens geformateerd uit te voeren. Als we het resultaat van voorbeeld TAB 1 bekijken, zien we dat op het eerste gezicht het scherm er regelmatig uitziet. Bij nadere bestudering blijkt echter dat van het rijtje getallen de eerste cijfers onder elkaar komen te staan zodat eenheden, tientallen, honderdtallen, enzovoort niet in dezelfde kolom staan. Zo kunt u niet goed zien of het om grote of kleine getallen gaat en wordt elke wiskundige bewerking een probleem. Dit is te verhelpen door aan TAB ook nog de grootte van het uit te voeren getal mee te geven. Gebruik daarvoor de instructie PRINT USING. De getallen in het voorbeeld beslaan hoogstens vier cijfers zodat in de uitvoer vier plaatsen moeten worden gereserveerd. Daarom voegen we de regel

```
PRINT USING "####"
```

toe (zie programma TAB 2). Nu staat alles netjes op een rijtje door de vruchtbare samenwerking tussen TAB en PRINT USING. Nog enkele opmerkingen.

- TAB is niet alleen bruikbaar op het scherm maar ook bij gebruik van een printer, en wel in combinatie met LPRINT.
- Als u na de vraag om INPUT in dezelfde regel met PRINT TAB verder wilt schrijven, moet u de cursor een regel terugzetten met

  ```
  LOCATE gewenste afstand,CSRLIN-1
  ```

  zoals te zien is in programma TAB 3. Als u met de TAB-instructie te weinig ruimte reserveert en de tekens elkaar overlappen, verdwijnen in SCREEN 0 en 1 de al aanwezige tekens maar op de grafische schermen SCREEN 2 en 3 blijven ze samen met de nieuwe tekens staan.

TAB 1

```
10 SCREEN 0
20 COLOR 1,15
30 WIDTH 40
40 FOR N=1 TO 20
50 PRINT INT(RND(1)*2000);
60 PRINT TAB(10);INT(RND(1)*2000);
70 PRINT TAB(20);INT(RND(1)*2000);
80 PRINT TAB(30);INT(RND(1)*2000)
90 NEXT N
```

TAB 2

```
10 SCREEN0
20 COLOR 1,15
30 WIDTH 40
40 FOR N=1 TO 20
50 PRINT USING "####";INT(RND(1)*2000);
60 PRINT TAB(10);USING "####";INT(RND(1)*2000);
70 PRINT TAB(20);USING "####";INT(RND(1)*2000);
80 PRINT TAB(30);USING "####";INT(RND(1)*2000)
90 NEXT N
```

TAB 3

```
10 SCREEN 0
20 WIDTH 40
30 COLOR 1,15
40 PRINT "getallen kleiner dan 100 invoeren"
50 PRINT
60 INPUT "getal ";A
70 IF A>99 THEN GOTO 60
80 LOCATE 24,CSRLIN-1
90 PRINT "kwadraat:";A*A
100 GOTO 60
```

3.6 LOCATE

De programma's LOCATE 1-11 staan aan het einde van deze paragraaf.

Met LOCATE kunt u in SCREEN 0 en 1 (1) een bepaalde schermpositie op een tekstscherm sturen en (2) de tekstcursor in- of uitschakelen.

Ad (1). Het is soms niet de bedoeling tekstuitvoer precies in de linkerbovenhoek van het scherm te laten beginnen (zie programma LOCATE 1). Met spaties en lege regels kunt u dat verhelpen, maar dat is geen handige oplossing (programma LOCATE 2). U ziet dat de listing onoverzichtelijk wordt, maar vervelender is dat een spatie een teken is en dus geheugenruimte in beslag neemt. Dat geldt in nog sterkere mate voor een hele regel spaties. Met de instructie TAB kunnen we de spaties (horizontaal) beter indelen (programma LOCATE 3), maar voor lege regels (verticaal) zouden we de instructie PRINT moeten gebruiken. Dat is niet effectief, zeker niet als we meerdere regels willen overslaan. Voor willekeurige sprongen, ook naar een andere regel, is er de instructie LOCATE. Deze instructie is vergelijkbaar met de instructie DRAW "B M 128,96" (voor de grafische schermen), die het middelpunt van het scherm aanspreekt. Er zijn twee parameters: voor de kolom (X) en voor de regel (Y). Houd wel in de gaten dat de telling van kolommen en regels begint bij 0 (0,0 is de linker bovenhoek van het scherm). Verder is de instelling van WIDTH van belang en de onderste regel (Y = 23 of 24). Het bereik van Y is weliswaar 0-31, maar bij grotere waarden van Y komt de uitvoer toch op de onderste regel te staan. Het bereik van X is 0-39. Bekijk voorbeeld LOCATE 4 of, iets overzichtelijker, LOCATE 5 maar eens. In tegenstelling tot PRINT zonder LOCATE (invoer van spaties, lege regels of TAB's) blijven met LOCATE alle tussenliggende tekens staan. Dat is duidelijk te zien in programma LOCATE 6. De puntkomma achter de PRINT-instructie is noodzakelijk; als die er niet staat, voert de MSX tekst uit inclusief line feed, waardoor de regel na de ingevoerde regel tot het einde zou worden gewist. Zo zouden we al aanwezige tekst wissen. Met een vooruitziende blik hebben we in het voorgaande programma de schrijfruimte beperkt tot regel 21 (dus 22 regels). Als regel 22 ook mee zou mogen doen (LOCATE 38,21), zouden de laatste letters van MSX op regel 22 komen te staan. Dat is op het scherm de 23e regel (de functiebalk). Daar kan niet op worden geschreven zodat het scherm automatisch scrollt en u de bovenste regel kwijtraakt. Voor zowel X als Y mag u waarden tot 256 nemen, maar de computer houdt slechts rekening met de op dat moment geldende maximale waarden. Bij WIDTH 40 in SCREEN 0 komt een teken dus niet verder dan kolom 39 ook al heeft u voor X de waarde 100 gekozen. In SCREEN 2 en 3 werkt de

instructie LOCATE niet. Bij deze grafische schermen moet u de instructie DRAW "B M X1,Y1" gebruiken.

Ad (2). Het kan heel hinderlijk om steeds de tekstcursor op het scherm te zien. Om deze uit te schakelen slaat u met komma's de eerste twee parameters van LOCATE over en neemt u voor de derde parameter een 0:

```
LOCATE ,,0
```

Weer inschakelen gaat dan met

```
LOCATE ,,1
```

of een andere waarde tot 256. Als de cursor uit is, heeft dat het voordeel dat u zelfs op de laatste schermregel nog kunt schrijven zonder dat het scherm automatisch (en waarschijnlijk tegen uw zin) scrollt. Alleen bij een lopend programma kan de cursor worden uitgeschakeld. Zodra u iets nieuws invoert (met INPUT en INPUT$) komt de cursor weer te voorschijn, of deze nu aan of uit was. De beide functies van LOCATE worden gedemonstreerd door programma LOCATE 7, waarin de cursor wordt uitgeschakeld en LOCATE 8, waarin hij wel zichtbaar is. Door deze aan en uit functie te koppelen aan een tijdlus (de subroutine ON INTERVAL in machinetaal) krijgen we een knipperende cursor (zie programma LOCATE 9).

Opmerkingen

- In de karakterset komt de cursor overeen met CHR$(255). Programma LOCATE 10 laat dat mooi zien. Eerst ziet u een donker vlak op het scherm, maar als u met de cursor naar de tekst gaat, verandert het vlak op het scherm meteen. Het ziet er nu uit als het omgekeerde van de letter waar de cursor staat. Eigenlijk is dat niets bijzonders: de waarde ofwel de betekenis van de cursor ligt in het VRAM. De actuele betekenis van CHR$(255) is afhankelijk van de plaats van de cursor op het scherm. In de meeste gevallen is de cursor omgekeerd aan een spatie.
- In plaats van met de instructie LOCATE kunnen we de X- en Y-coördinaat ook sturen door de gewenste waarden rechtstreeks in het schermgeheugen te POKEn. Het gaat hierbij om de adressen van de systeemvariabelen.

&HF3DC	Y-positie
&HF3DD	X-positie

De instructie LOCATE 15,10 komt dan overeen met

```
POKE &HF3DD,15
POKE &HF3DC,10
```

Zo kunt u ook de cursor aan of uit zetten.

```
POKE &HFCA9,0      cursor uit
POKE &HFCA9,1      cursor aan
```

Met PEEK en POKE kunt u in principe elke BASIC-instructie rechtstreeks in het geheugen uitvoeren. BASIC heeft wel het voordeel dat het erg simpel is en dat het sneller werkt dan wanneer u duizenden adressen moet onthouden en aangeven.

- Met LOCATE X,Y kunt u een willekeurige schermpositie aanspreken en er iets op schrijven. U kunt ook alle posities doornummeren. Dan moet u wel direct in het VRAM werken (zie programma LOCATE 11). Midden op het scherm verschijnt nu een verticale uitroepteken-lijn (CHR$(33)) tot onderaan het scherm, die regel voor regel is opgebouwd met STEP 40.

LOCATE 1
```
10 SCREEN 0
20 COLOR 1,15
30 PRINT "MENU"
40 PRINT "1. adressen invoeren"
50 PRINT "2. adressen zoeken"
60 PRINT "3. adressen opslaan"
```

LOCATE 2
```
10 SCREEN 0
20 COLOR 1,15
30 PRINT"      MENU"
31 PRINT
40 PRINT"      1. adressen invoeren"
50 PRINT"      2. adressen zoeken"
60 PRINT"      3. adressen opslaan"
```

LOCATE 3
```
10 SCREEN 0
20 COLOR 1,15
30 PRINT TAB(15) "MENU"
31 PRINT
40 PRINT TAB(10) "1. adressen invoeren"
50 PRINT TAB(10) "2. adressen zoeken"
60 PRINT TAB(10) "3. adressen opslaan"
```

LOCATE 4
```
10 SCREEN 0
20 COLOR 1,15
30 LOCATE 15,5
40 PRINT "MENU"
50 LOCATE 10,7
60 PRINT "1. adressen invoeren"
70 LOCATE 10,8
80 PRINT "2. adressen zoeken"
90 LOCATE 10,9
100 PRINT "3. adressen opslaan"
```

LOCATE 5
```
10 SCREEN 0
20 COLOR 1,15
30 LOCATE 15,5
40 PRINT "MENU"
50 FOR N=7 TO 9
60 LOCATE 10,N
70 READ A$
80 PRINT A$
90 NEXT N
100 DATA 1. adressen invoeren,2. adressen zoeken,
3. adressen opslaan
```

LOCATE 6
```
10 SCREEN 0
20 COLOR  1,15
30 FOR N=1 TO 100
```

```
40 LOCATE INT(RND(1)*39),INT(RND(1)*21)
50 PRINT "MSX";
60 NEXT N
```

LOCATE 7

```
10 SCREEN 0
20 COLOR 1,15
30 LOCATE 0,0,0
40 PRINT "MSX"
50 GOTO 50
```

LOCATE 8

```
10 SCREEN 0
20 COLOR 1,15
30 LOCATE 0,0,1
40 PRINT "MSX"
50 GOTO 50
```

LOCATE 9

```
10 SCREEN 0
20 COLOR 1,15
30 ON INTERVAL=10 GOSUB 60
40 INTERVAL ON
50 GOTO 50
60 IF Z=1 THEN Z=0 ELSE Z=1
70 ON Z+1 GOTO 80,100
80 LOCATE ,,1
90 RETURN
100 LOCATE ,,0
110 RETURN
```

LOCATE 10

```
10 WIDTH 40
20 SCREEN 0
30 COLOR 1,15
40 FOR N=1 TO 960
50 PRINT CHR$(255);
60 NEXT N
70 PRINT "MSX is fantastisch"
```

LOCATE 11

```
10 WIDTH 40
20 SCREEN 0
30 COLOR 1,15
40 FOR N=20 TO 940 STEP 40
50 VPOKE N,33
60 NEXT N
```

3.7 POS

De programma's POS 1-6 staan aan het einde van deze paragraaf.

Er zijn diverse mogelijkheden om informatie te verkrijgen over de positie van de cursor op een tekstscherm. In de vorige paragraaf bespraken we de instructie LOCATE die bijvoorbeeld bruikbaar is om aan te geven waar een tekst op het scherm moet komen (PRINT) of waar nieuwe invoer heen moet (INPUT). Met CSRLIN leest u de waarde van de systeemvariabele die het regelnummer van de cursor aangeeft. Dat is dus zijn Y-waarde. Zo is er voor de X-waarde de systeemvariabele POS. Onderzoek het effect ervan met voorbeeld POS 1. Het scherm komt vol te staan met de afkorting MSX. Daarna verschijnt een getal dat de actuele tekenpositie aangeeft. Wilt u ook nog weten op welke regel u zich bevindt, dan dient in het programma een regel te staan die met CSRLIN de betreffende regel uitleest.

```
45 PRINT CSRLIN;
```

Als we de X- en Y-coördinaat op een tekstscherm kennen, wat hebben we daar dan aan? Twee toepassingen; eerst programma POS 2. We willen graag dat het resultaat voor iedere MSX-gebruiker hetzelfde is. Daarom moeten eerst de schermvariabelen worden bepaald. De tekst in regel 50 loopt tot kolom 35 (POS(0)). Als de in te voeren naam langer is dan de nog resterende vrije ruimte (40-35=5 tekens), dan gaat het programma met een extra PRINT-instructie eerst naar de volgende regel en neemt de hele naam mee (word wrap). De lengte van de naam wordt berekend met LEN(A$). De tweede toepassing is het formateren van uitvoer op het scherm (zie programma POS 3). De voorwaarde voor uitvoer van de letters MSX is dat de cursorpositie een veelvoud van 4 is. Zoiets kunt u ook toepassen bij de uitvoer van een tabel of een spreadsheet. Met alleen POS komen de linkerkanten van de getallen onder elkaar, voor berekeningen erg onhandig (zie programma POS 4). Om dit echt mooi uit te voeren moet u ook de instructie PRINT USING gebruiken: dan komen de rechterkanten van de getallen onder elkaar, zoals u in programma POS 5 kunt zien. De waarde van de variabele CSRLIN kunt u direct met PRINT op het scherm krijgen. Aan POS moet u nog een willekeurig getal toevoegen voor u de waarde ervan kunt oproepen. De instructie moet de vorm POS(N) hebben. Omdat de systeemvariabelen POS(N) en CSRLIN overeenkomen met respectievelijk X en Y, kunt u ook rechtstreeks de betreffende adressen in het RAM oproepen.

```
POS(N)      &HF3DD
CSRLIN      &HF3DC
```

Let er wel op dat bij de systeemvariabelen vanaf 0 wordt geteld en

bij bovengenoemde RAM-adressen vanaf 1. Dit soort instructies bestaat niet voor de grafische schermen. Natuurlijk staan de diverse posities wel in het geheugen.

coördinaat	geheugenplaats
X (absoluut)	&HFCB3
Y (absoluut)	&HFCB5
X (relatief)	&HFCB7
Y (relatief)	&HFCB9

Voorbeeld POS 6 laat een toepassing zien. Het resultaat is afhankelijk van regel 30. Ter vergelijking staat hieronder ook het resultaat van POS 6 met een andere regel 30.

geheugenplaats	DRAW"N R8"	CIRCLE(128,96)20
&HFCB3	136	20
&HFCB5	96	96
&HFCB7	128	128
&HFCB9	96	96

Bij absolute posities (absolute adressering) gaan we uit van de oorsprong van het beeldscherm, maar bij relatieve (relatieve adressering) van een ander punt op het scherm. Bij de instructie CIRCLE bijvoorbeeld geeft u met STEP aan hoe groot de straal van de cirkel moet zijn. Het middelpunt dient in dat geval als referentiepunt voor de tekening en niet de oorsprong van het beeldscherm. Maar die oorsprong is natuurlijk wel de maatstaf voor de plaats van het middelpunt. De instructie LOCATE X,Y voor de tekstschermen heeft als tegenhangers voor de grafische schermen DRAW "B M X,Y" of PRESET(X,Y). Met veel grafische instructies is het mogelijk (tijdelijk) buiten de grenzen van het scherm te werken. Op zo'n moment levert POS(N) de waarde -1 op. Hoewel POS(N) niet zo ontzettend handig is bij de grafische schermen, krijgt u toch informatie over het kolomnummer van de cursor in het laatst gebruikte tekstscherm.

POS 1

```
10 SCREEN 0
20 COLOR 1,15
30 PRINT "MSX";
40 PRINT POS(1);
50 GOTO 30
```

POS 2

```
10 SCREEN 0
20 COLOR 1,15
30 WIDTH 40
40 INPUT "Uw naam alstublieft         ";A$
50 PRINT "Het doet mij bijzonder veel genoegen
 dat ik u begroeten kan: meneer/mevrouw ";
60 IF LEN(A$)>40-POS(0) THEN PRINT:PRINT A$ ELSE PRINT A$
```

POS 3

```
10 SCREEN 0
20 COLOR 1,15
30 WIDTH 40
40 FOR N=1 TO 1000
50 IF POS(1)/4=INT(POS(1)/4) THEN PRINT "MSX ";
60 NEXT N
```

POS 4

```
10 SCREEN 0
20 COLOR 1,15
30 WIDTH 40
40 FOR N=1 TO 1000
50 IF POS(1)/5=INT(POS(1)/5) THEN PRINT
 INT(RND(1)*999);ELSE PRINT " ";
60 NEXT N
```

POS 5

```
10 SCREEN 0
20 COLOR 1,15
30 WIDTH 40
40 FOR N=1 TO 1000
50 IF POS(1)/5=INT(POS(1)/5) THEN PRINT USING
 "###";INT(RND(1)*999);ELSE PRINT " ";
60 NEXT N
```

POS 6

```
10 SCREEN 2
20 PSET(128,96)
30 DRAW"n r8"
40 A=PEEK(&HFCB3)
50 B=PEEK(&HFCB5)
60 C=PEEK(&HFCB7)
70 D=PEEK(&HFCB9)
80 SCREEN0
90 PRINT A;B;C;D
100 PRINT
110 LIST
```

3.8 LPOS

Het programma LPOS 1 staat aan het einde van deze paragraaf.

LPOS (line printer position) laat de positie (kolom) zien waar de matrixkop of het margrietwiel van de printer zich op dat moment bevindt. Dat is belangrijk om bij de uitvoer van tabellen de gegevens netjes onder elkaar te kunnen krijgen. Programma LPOS 1 demonstreert deze functie zowel op het scherm als op de printer. Op het scherm (POS(1)) en op papier (LPOS(1)) verschijnen driemaal twee getallen die in een tabel onder elkaar staan. U kunt LPOS(1) ook gebruiken om bij een uitgevoerde tekst de printer te vragen of er op een bepaalde regel nog genoeg ruimte is om het volgende woord af te drukken. Zoniet, dan moet met LPRINT een line feed worden gegeven.

Opmerkingen

- Het is natuurlijk ook mogelijk de positie van de printerkop rechtstreeks uit het geheugen te lezen:

  ```
  PRINT &HF415
  ```

  Behalve dit adres lezen kunt u er ook een waarde aan toekennen, zodat de printerkop naar de betreffende kolom toegaat. Dat heeft iets weg van een LOCATE-instructie.

  ```
  POKE &HF415,40
  ```

- Om de printerkop te sturen kunt u beter niet de methode in de vorige opmerking gebruiken. Dat kan veel beter met LPRINT TAB, LPRINT USING of zelfs een combinatie daarvan: LPRINT USING TAB. Voorbeelden staan in paragraaf 3.5. Deze opmerking geldt niet voor de controle van de printerkop aan het eind van een regel, waar steeds wordt bekeken of er nog genoeg ruimte over is om een nieuw woord te printen.

LPOS 1

```
10 FOR N=1 TO 3
20 READ A,B
30 IF POS(1)<>5 THEN PRINT " ";:GOTO 30
40 PRINT A
50 IF LPOS(1)<>5 THEN LPRINT " ";:GOTO 50
60 LPRINT A;
70 IF POS(1)<>20 THEN PRINT " ";:GOTO 70
80 PRINT B;
90 IF LPOS(1)<>20 THEN LPRINT " ";:GOTO 90
100 LPRINT B;
110 PRINT
120 LPRINT
130 NEXT N
140 DATA 10,20,30,40,50,60
```

3.9 CSRLIN

Het programma CSRLIN 1 staat aan het einde van deze paragraaf.

Met de instructie CSRLIN komt u te weten in welke regel de cursor staat (in SCREEN 0 en SCREEN 1). Zo kunt u voorkomen dat u over een al aanwezige tekst heenschrijft. Stel dat op regel 9 de afkorting MSX staat en dat de rest van het scherm moet worden volgeschreven met het woord SUPER, dan moet de plaats van de cursor wel goed worden bijgehouden. Let erop dat de computer niet telt van 1 tot en met 10 maar van 0 tot en met 9 (zie programma CSRLIN 1). Pas als de computer de zekerheid heeft dat er nog niets op een regel staat (IF CSRLIN 9) gaat hij verder met de volgende opdracht (THEN PRINT "SUPER"). De instructie CSRLIN is alleen te gebruiken voor de regels 0-22 van SCREEN 0 en 1; eventueel ook in regel 23 als u de functiebalk niet gebruikt. Als een tekst eenmaal buiten het schermkader is gescrolled is hij onherroepelijk verloren. CSRLIN is pas echt zinvol als u bijvoorbeeld zelf een tekstverwerkingsprogramma heeft geschreven. Naast CSRLIN voor de Y-positie heeft u dan ook POS nodig voor de X-positie en LOCATE om de cursor op het beeldscherm te kunnen sturen. Een opmerking tot slot. We kunnen de Y-positie van de cursor ook rechtstreeks uit het geheugen oproepen, en wel op adres &HF3DC. Voor de grafische schermen is dat adres &HFCB5. Bovendien hebben we bij de grafische schermen de relatieve positie ten opzichte van de oorsprong (zie paragraaf 3.7). Die staat op adres &HFCB9.

CSRLIN 1

```
10 SCREEN 0
20 COLOR 1,15
30 LOCATE 0,9
40 PRINT "MSX"
50 FOR M=1 TO 1000
60 NEXT M
70 LOCATE 0,0
80 FOR N=0 TO 20
90 IF CSRLIN<>9 THEN PRINT "fantastisch"
 ELSE LOCATE 0,CSRLIN+1
100 NEXT N
```

4 De karakterset van de MSX

4.1 CHR$

De programma's CHR$ 1-5 staan aan het einde van deze paragraaf.

Alle karakters zijn in het geheugen opgeslagen als getallen. Elk karakter heeft zijn eigen code, de zogenaamde ASCII-code. Met de CHR$-instructie komt u aan de weet welk karakter er bij een bepaalde ASCII-code hoort. Omgekeerd is er de instructie ASC om de code te vinden bij een bepaald karakter (zie paragraaf 4.2). Bijvoorbeeld

```
PRINT CHR$(65)
```

resultaat: A

Zo kunt u de hele karakterset van de MSX op het scherm krijgen. Programma CHR$ 1 laat dat zien. Eerst verschijnen er een paar interpunctietekens (punt, komma, ...), dan de cijfers 0-9, de hoofdletters en de kleine letters, een aantal internationale buitengewone tekens en tot slot een paar wiskundige en grafische tekens. We hebben nu lang niet alle ASCII-codes opgeroepen. De codes 0-32 zijn dan ook niet zichtbaar. Sommige codes zijn signalen. Neem bijvoorbeeld

```
PRINT CHR$(7)
```

Als u als monitor een t.v. gebruikt, hoort u na deze invoer een PIEP (zie ook BEEP, paragraaf 6.4) en op het scherm verschijnt de melding Ok. Ook met

```
PRINT CHR$(13)
```

komt er geen teken op het scherm maar een lege regel zodat de cursor een regel zakt. Hiermee simuleert u dus de RETURN-functie. Het nut van CHR$(7) bewijst programma CHR$ 2. Als u zich in dit programma aan de aanwijzingen houdt, gebeurt er niets vreemds. Maar als u de in regel 30 aangegeven grens overschrijdt, volgen de twee reacties die in regel 40 staan. CHR$(13) kan goed worden gecombineerd met een instructie. RETURN houdt in dat de eraan voorafgaande instructie wordt uitgevoerd. We gebruiken deze kennis om de functietoetsen F1 en F2 opnieuw te definiëren.

```
KEY 1,"LIST"
KEY 2,"LIST"+CHR$(13)
```

Even aangenomen dat er nog een programma in het geheugen zit, wordt nu duidelijk dat alleen F1 niets oplevert; er is ook nog een RETURN nodig. F2 zorgt ervoor dat de instructie gelijk wordt uitgevoerd, dankzij de toevoeging CHR$(13). Er is nog een BASIC-instructie waarbij u niet om de instructie CHR$ heen kunt, namelijk SPRITE$. Als u geen al te hoge eisen stelt (u wilt bijvoorbeeld een sprite die uit een rechte lijn bestaat), kunt u die direct invoeren via het toetsenbord.

```
SPRITE$(0)="UUUUUUUU"
```

Als we deze sprite bijvoorbeeld willen splitsen door twee lege regels in te voeren, zouden we een aantal keren CHR$(0) moeten intypen. Veel eenvoudiger is het om SPRITE$ uit te breiden met de instructie CHR$.

```
SPRITE$(0)="UUU"+CHR$(0)+CHR$(0)+"UUU"
```

Er zijn heel wat karakterreeksen die u rechtstreeks via het toetsenbord kunt invoeren, maar die in een programma niet zo makkelijk terug te vinden zijn. Zo wordt in het bedrijfssysteem CP/M, dat loopt op een MSX met diskdrive, veel gebruik gemaakt van de combinatie CTRL + letter. Met behulp van CHR$ is het niet moeilijk om in een BASIC-programma na te gaan of iemand een combinatie met CTRL heeft gebruikt. Programma CHR$ 3 laat dat zien. Dit programma reageert alleen op de combinatie van CTRL met een letter. Voert u bijvoorbeeld CTRL-E in (wissen tot het eind van de regel), dan ziet u op het scherm of de computer het heeft begrepen. Verder kunt u bij veel printers met CHR$ kiezen uit verschillende lettertypen. Bij Epson printers schakelt u de aansturing van de printerkop in met de ESC-toets, die overeenkomt met CHR$(27). Voor vet drukken (emphasized mode) is er bijvoorbeeld de instructie

```
PRINT CHR$(27);"E"
```

Van de vele andere mogelijkheden van de CHR$-instructie heeft u er na bestudering van dit hoofdstuk vast al een aantal zelf ontdekt. Een ervan zullen we nader bekijken. Soms moet u in een programma hoofdletters invoeren, maar u weet niet zeker of de MSX dat ook daadwerkelijk doet. Programma CHR$ 4 verandert kleine letters in hoofdletters. Of u nu kleine letters of hoofdletters invoert, elke tekst verschijnt in hoofdletters op het scherm. Het bereik van CHR$ is 0-255. Overschrijdt u dit, dan volgt de foutmelding Illegal function call. CHR$(255) komt overeen met het cursorteken. Dat kan een leuk effect opleveren, zoals programma CHR$ 5 laat zien. De cursor wordt een aantal keren op het scherm weergegeven. Zodra de cursor op een bepaalde plaats komt te staan, verandert meteen het cursorteken. Vandaar de snelle

wisseling van het hele beeldscherm, hoewel die uiteindelijk alleen maar te maken heeft met de snel wisselende weergave van slechts 1 karakter, namelijk de cursor. Vergelijk dit programma eens met LOCATE 10 in paragraaf 3.6.

CHR$ 1
```
10 COLOR 1,15
20 SCREEN 0
30 WIDTH 40
40 FOR N=32 TO 255
50 PRINT CHR$(N);
60 NEXT N
```

CHR$ 2
```
10 COLOR 1,15
20 SCREEN 0
30 WIDTH 40:INPUT "voer een getal kleiner
 dan 100 in";A
40 IF A>99 OR A<0 THEN PRINT"goed op mijn
 aanwijzingen letten";CHR$(7)
50 GOTO 30
```

CHR$ 3
```
10 CLS
20 A$=INKEY$
30 IF A$="" THEN GOTO 20
40 IF ASC(A$)<32 THEN GOTO 50 ELSE GOTO 20
50 PRINT "U hebt CTRL-";CHR$(ASC(A$)+64);" ingedrukt"
60 GOTO 20
```

CHR$ 4
```
10 SCREEN 0
20 COLOR 1,15
30 WIDTH 40
40 Z=96
50 INPUT "voer een woord in ";A$
60 FOR N=1 TO LEN(A$)
70 IF MID$(A$,N,1)=>"a" THEN MID$(A$,N,1)
=CHR$(ASC(MID$(A$,N,1))-32)
80 NEXT N
90 PRINT A$
```

CHR$ 5
```
10 SCREEN 0
20 COLOR  1,15
30 WIDTH 40
40 FOR N=1 TO 23
50 PRINT STRING$(40,255);
60 NEXT N
70 PRINT "zet de cursor op een letter"
```

4.2 ASC

De programma's ASC 1-3 staan aan het einde van deze paragraaf.

De instructie ASC is het omgekeerde van CHR$ en levert de decimale code van een karakter.

```
10 PRINT ASC("A")

resultaat: 65
```

Om ervoor te zorgen dat de MSX de A niet als variabele of als losse rommel beschouwt, moet deze tussen aanhalingstekens staan; de A wordt dan als string opgevat. Bij een langere string is het resultaat van bovenstaande regel dat alleen het eerste teken telt.

```
10 A$ = "Hottentottententententoonstelling"
20 PRINT ASC(A$)

resultaat: 72
```

Met een kleine verandering kan het resultaat worden omgezet in binaire vorm.

```
10 A$ = "Hottentottententententoonstelling"
20 PRINT BIN$(ASC(A$)

resultaat: 1001000
```

Toepassingen van ASC.

- We kunnen toetsen herkennen die op het scherm niet direct kunnen worden weergegeven (bijvoorbeeld de besturingstoetsen). Zonder speciale maatregelen is het niet zichtbaar of iemand na een invoer heeft afgesloten met RETURN. Dat geldt ook voor de instructie INPUT$ (zie programma ASC 1). Het resultaat is een lege regel en dat is dus niet te zien. Kijken we naar de lengte van A$, dan zien we dat er wel degelijk een teken is ingevoerd.

  ```
  PRINT LEN(A$)

  resultaat: 1
  ```

  Met ASC gaan we de betekenis van het ingevoerde teken na (programma ASC 1a). Resultaat: 13. Dat is dus blijkbaar de code voor de RETURN-toets. U kunt nu een programma zo veranderen dat het pas verder gaat nadat u op RETURN hebt

gedrukt (zie programma ASC 1b).

- Met behulp van ASC kunnen we grafische tekens ook op een andere dan een MSX printer uitvoeren. Omdat de karakterset van de MSX enigszins afwijkt buiten de sector van de codes 32 (spatie) tot 126 (slangetje) hebben de meeste printers daar moeite mee. U moet eerst eens controleren of uw printer uw MSX goedgezind is.

```
10 SCREEN ,,,,0
20 LPRINT " "  GRAPH en P tegelijk indrukken
tussen de aanhalingstekens
```

Als op de printer ook het ingevoerde zwarte blokje verschijnt is alles in orde en kunt u doorlezen bij punt 3). Maakt uw printer er wat anders van dan is het noodzakelijk om door middel van

```
SCREEN ,,,,1
```

het sein te geven om alle grafische tekens in het vervolg als spaties uit te voeren. Als u een listing naar een MSX-makker of een tijdschrift stuurt moet u de grafische MSX-tekens naderhand intekenen of in plaats daarvan de codes afdrukken. Programma ASC 2 kan maximaal 255 opeenvolgende tekens coderen. In plaats van grafische tekens, waar anti-MSX-printers niets mee kunnen aanvangen, zet u nu in de listing:

```
LPRINT CHR$(180);CHR$(...);CHR$(...); ...enzovoort
```

In paragraaf 4.1 gaven we al aan dat zo al de gewenste tekens weer te voorschijn zullen komen.

Het is mogelijk het uiterlijk te veranderen van een teken dat direct via het toetsenbord bereikbaar is. Door met de instructie ASC de ASCII-code van het karakter te achterhalen kunnen we vervolgens te weten komen op welk adres in het VRAM dat teken begint. Programma ASC 3 levert die informatie. Het adres van de spatie is 2304. Er zijn 8 adressen nodig voor de definitie van een teken; hier dus de nummers 2304-2311. U kunt daar een ander teken neerzetten:

```
VPOKE 2304,255
```

Het hele scherm ziet er daarna anders uit en dat wordt nog duidelijker door CLS in te voeren. Dit verschil van dag en nacht heeft als reden dat het beeldscherm bijna alleen uit spaties bestaat. U kunt de betovering op twee manieren verbreken. Met een SCREEN-instructie krijgen alle gegevens in het VRAM hun startwaarden weer terug, zodat u alle

veranderingen kwijt bent. Maar een extra POKE-instructie voldoet ook prima:

```
VPOKE 2304,0
```

Twee opmerkingen tot besluit. Omdat de tekencodes lopen van 0-255 zult u met ASC nooit getallen buiten dat gebied terugkrijgen. In het verlengde hiervan ligt de volgende waarschuwing: gebruik geen lege strings bij ASC.

```
10 A$=""
20 PRINT ASC(A$)
```

De computer antwoordt met Illegal function call; er valt immers niets te halen.

ASC 1
```
10 COLOR 1,15
20 SCREEN 0
30 PRINT "RETURN-toets indrukken"
40 A$=INPUT$(1)
50 PRINT A$
```

ASC 1a
```
10 COLOR 1,15
20 SCREEN 0
30 PRINT "RETURN-toets indrukken"
40 A$=INPUT$(1)
50 PRINT ASC(A$)
```

ASC 1b
```
10 COLOR 1,15
20 SCREEN0
30 PRINT "RETURN-toets indrukken"
40 A$=INPUT$(1)
50 IF ASC(A$)=13 THEN PRINT "klaar" ELSE GOTO 30
```

ASC 2
```
10 COLOR 1,15
20 SCREEN 0
30 INPUT "voer iets in ";A$
40 FOR N=1 TO LEN(A$)
50 PRINT MID$(A$,N,1);" = ";: PRINT ASC(MID$(A$,N,1)),
60 NEXT N
```

ASC 3
```
10 COLOR 1,15
20 SCREEN 0
30 PRINT "spatie = CHR$(32)"
40 START=2048+32*8
50 PRINT START
```

5 Grafische instructies

5.1 DRAW

De programma's DRAW 1-14 staan aan het einde van deze paragraaf.

DRAW is meer dan alleen een instructie, maar minder dan een echte grafische taal als LOGO. Bij DRAW horen een hele reeks subcommando's, allemaal om lijnen te tekenen. De mogelijkheden hiervan zijn beperkt. Alleen Turtle Graphics, een onderdeel van LOGO, kan met DRAW worden nagebootst. Microsoft, de ontwerper van MSX-BASIC, heeft het bij deze instructie over GML (Graphics Macro Language, dit in tegenstelling tot grafiek per pixel). DRAW is een grafische instructie en kan dus alleen worden ingezet op de grafische schermen. In SCREEN 0 of SCREEN 1 reageert de MSX met Illegal function call. Programma DRAW 1 is de eerste kennismaking met DRAW. De aanhalingstekens achter DRAW geven al aan dat het om een string gaat; ook stringoperaties zijn mogelijk.

```
DRAW "B M 128,96 ..."
```

Dit deel van de instructie zorgt ervoor dat de cursor zonder een lijn te trekken (de B van blind) gaat naar (de M van move) positie 128,96. Het gedeelte daarna regelt verplaatsingen van de cursor waarbij wel een lijn wordt getrokken. Hierbij maken we gebruik van de Engelse richtingaanduidingen.

```
U = up    = naar boven
D = down  = naar beneden
L = left  = naar links
R = right = naar rechts
```

U hoeft alleen nog een getal toe te voegen om aan te geven waar de cursor naartoe moet.

```
DRAW "B M 128,96 U8 R8 U8 R8 D16 L16"
```

Het resultaat is een trapje. Om de instructie makkelijker te kunnen lezen, hebben we tussen de onderdelen spaties gezet. Die mag u weglaten of vervangen door puntkomma's. Alleen als het gaat om twee verschillende waarden voor de schermcoördinaten (X en Y), moet u die scheiden door een komma. Er zijn ook aanduidingen voor tussenliggende richtingen. Daarvoor gebruiken we vier opeenvolgende letters. De draairichting is met de klok mee.

```
E = noordoost
```

F = zuidoost
G = zuidwest
H = noordwest

U kent vast nog wel de truc om een huis te tekenen zonder het potlood van het papier te nemen of een lijn dubbel te trekken. Probeer nu eens met een DRAW-instructie hetzelfde te bereiken. Programma DRAW 2 geeft een mogelijke oplossing. Om het huis tweemaal zo groot te maken, hoeft u alleen de getallen te verdubbelen. In dat geval luidt regel 30:

```
30 DRAW "B M 128,96 R16 U16 L16 E8 F8 G16 U16 F16"
```

De ontwerpers van MSX-BASIC vonden deze methode te ingewikkeld. U kunt ook uitgaan van de oorspronkelijke vorm en vervolgens een schaalfactor gebruiken. Om de vermenigvuldigingsfactor te vinden, deelt u de schaalfactor door 4. Het symbool S4 geeft aan dat de grootte niet verandert. Voor een verdubbeling voert u dus S8 in en voor een halvering S2. U kunt waarden van 1 tot 256 invoeren. Door een grote schaalfactor overschrijdt u al gauw de grenzen van het scherm. Ver over de rand liggende lijnen worden afgebeeld aan de rand van het scherm. De verhoudingen kloppen dan niet meer. Programma DRAW 3 laat dat zien. U kunt de uitvoering van het programma stoppen met de toetsencombinatie CTRL/STOP en hervatten met RUN. Vreemd genoeg is dan het oorspronkelijke vierkant verdwenen en is alleen het te grote vierkant overgebleven. Net als bij de parameters van de instructie COLOR blijft de waarde van de schaalfactor geldig (aanwezig in het RAM) tot de invoer van een nieuwe waarde. Als u weer terug wilt naar de oorspronkelijke grootte (S4), moet u dat expliciet aangeven.

```
30 DRAW "B M 128,96 S4 R8 U8 L8 D8"
```

DRAW kent verder de parameter A, die een draaiingshoek (angle) rechtsom aangeeft. U kunt hoeken kiezen in stappen van 90 graden. Eerst maken we een gewone tekening: programma DRAW 4. Voeg nu aan regel 30 een draaiing over 90 graden toe.

```
30 DRAW "B M 128,96 A1 R8 U8 R8"
```

Niet alleen is de hele trap gedraaid, maar ook de verschillende verplaatsingsinstructies. Naar rechts verandert in naar beneden, naar links wordt naar boven, enzovoorts. Dat kan voor alle hoeken worden samengevat in een schema.

A0 (0 grad.)	U	R	D	L
A1 (90 grad.)	R	D	L	U
A2 (180 grad.)	D	L	U	R

A3 (270 grad.)   L   U   R   D
A4 (360 grad.) oftewel A0

Met een korte subroutine kunt u figuren met DRAW over een willekeurige hoek verdraaien. Parameter A is handig als u figuren wilt maken waarbij een bepaalde vorm in verschillende richtingen terugkeert. Programma DRAW 5 tekent een kompas. De DRAW-parameters uit regel 30 blijven hetzelfde en dank zij A1, A2, enzovoort verschijnt er om de 90 graden een pijl. Zo kunt u zich de moeite besparen alle pijlen apart te tekenen. Er is nog een handige toepassing van deze parameter: met behulp van een programma dat verderop in het boek staat, kunt u met DRAW karakters op een grafisch scherm afbeelden. Met parameter A kunt u dan natuurlijk ook grafische afbeeldingen verfraaien met verticaal of ondersteboven staande tekst. DRAW kent ook parameter N, die ervoor zorgt dat het programma na een tekeninstructie terugkeert naar het uitgangspunt. Om dit te demonstreren vereenvoudigen we het eerder getekende kompas door de pijlpunten weg te laten (zie programma DRAW 6). Dit programma verplaatst de cursor naar het midden van het scherm (128,96), tekent een lijn (U8) in vier verschillende richtingen (A0-A3) en keert daarbij steeds terug naar het middelpunt van het scherm (N). U kunt het plaatje vergroten door een schaalfactor toe te voegen (hier S12). Het uiteindelijke resultaat is een windroos.

```
30 DRAW "B M 128,96 S12 N A0 U8 N A1 U8 N A2 U8 N A3 U8"
```

De instructies DRAW "M" en DRAW "B M" zijn al eerder ter sprake gekomen. Met M (move), gevolgd door twee coördinaten, beweegt de cursor vanaf de actuele naar de aangegeven schermpositie en tekent daarbij een lijn in de gewenste kleur. Voegt u nog B (blind) toe, dan gebeurt hetzelfde zonder dat er een lijn wordt getrokken. DRAW "B M" kan bij de grafische schermen dezelfde functie vervullen als LOCATE bij een tekstscherm. Met deze instructie kunt u punt voor punt grafische afbeeldingen of teksten op het scherm plaatsen. Een andere toepassing hiervan heeft betrekking op de manier van adresseren: posities op een scherm kunnen absoluut en relatief worden aangegeven. Bij het absolute systeem is het uitgangspunt de linkerbovenhoek (0,0). De rechter benedenhoek heeft de coördinaten 256,192. Het middelpunt van het scherm is dus 128,96. Bij een relatieve plaatsaanduiding is het uitgangspunt de voorgaande positie van de grafische cursor. We verplaatsen deze met STEP, gevolgd door positieve of negatieve waarden (afhankelijk van het uitganspunt) voor de coördinaten. Door voor alle waarden een + of een - te zetten, weet de MSX direct dat het om relatieve adressering gaat. Met de functie STEP kunt u hele tekeningen op een andere plaats op het scherm herhalen. Dat laat het volgende programma (DRAW 7) zien. Eenvoudig, nietwaar? Het is ook niet moeilijk om de vierkantjes

op een diagonale lijn te zetten in plaats van naast elkaar.

```
50 DRAW "B M +8,+8 R4 U4 L4 D4"
```

Het belang van relatieve adressering wordt wel duidelijk als u meer gecompliceerde tekenopdrachten wilt uitvoeren. Een goed voorbeeld is de karakterset van DRAW, die alleen werkt met relatieve adressering. De letters hebben onderling dezelfde afstand. Met de absolute plaatsbepaling zouden we ze stuk voor stuk op het scherm moeten plaatsen, gerekend vanuit positie 0,0. Een ander voorbeeld zijn de schaakstukken op een schaakbord. Door de relatieve adressering is het geen kunst zestien fraaie pionnen te tekenen (zie programma DRAW 8). Hopelijk zijn nu ook de twijfelaars overtuigd van de fascinerende mogelijkheden van de instructie DRAW. De laatste parameter van DRAW is C (colour), waarmee de kleur wordt aangegeven. Na C volgt het kleurgetal (zie voor de kleurcodes paragraaf 3.4). In het volgende voorbeeld (DRAW 9) gaan we eerst met DRAW "S4 A0" terug naar de normale toestand. Nadat de grafische cursor is gecentreerd (op positie 128,96) geeft C15 aan dat er met de kleur wit wordt getekend en wel over een afstand van 16 punten naar rechts. Daarna wisselt de kleur iedere keer als de richting verandert, zodat tenslotte het getekende vierkant twee witte en twee zwarte zijden heeft. Hetzelfde resultaat is te verkrijgen door tussen de tekeninstructies steeds van voorgrondkleur te veranderen met COLOR. Hier is een waarschuwing op zijn plaats, omdat kleurveranderingen eveneens buiten de DRAW-instructie gelden, zelfs na wisseling van schermmodus! Laten we toch maar eens kijken hoe het voorbeeld er met COLOR uit komt te zien (programma DRAW 10). Hoewel het programma nu knap lang is geworden, doet het niets meer dan DRAW 9. Eigenlijk moet de regel

```
COLOR 15
```

nog worden toegevoegd om de uitgangskleur weer terug te krijgen. Hoewel de MSX werkt met 256*192 grafische punten, zijn er voor de kleuren maar 32*192 'kleurstreepjes' beschikbaar. Maak de gekleurde stukjes niet al te klein, anders gaan de kleuren door elkaar heenlopen zoals in voorbeeld DRAW 11. Eigenlijk zouden we nu net als in DRAW 9 steeds twee witte en twee zwarte lijnen moeten zien, maar daar komt niets van terecht omdat op veel plaatsen wit, zwart en de achtergrondkleur dicht bij elkaar zitten; dat is minstens een kleur te veel. Het resultaat is een aantal kleurvlekken.

Samengevat heeft DRAW de volgende parameters:

U R D L	aanduiding 90 graden-richtingen
E F G H	aanduiding 45 graden-richtingen

A	draaiing om een hoek, per 90 graden
M	cursor verplaatsen en lijn trekken
B M	cursor verplaatsen zonder lijn te trekken
C	kleur bepalen
S	schaal (vergroting of verkleining)
M 128,96	absolute adressering
M +50,-50	relatieve adressering

Variabelen binnen DRAW

Een parameter (p) van DRAW kan als variabele worden gedefinieerd.

p = variabelenaam ;

De aanduiding als variabele alleen is niet genoeg; hij moet altijd tussen = en puntkomma staan. Is dat niet het geval, dan interpreteert de MSX bijvoorbeeld een A als de instructie A, dus een draaiing. Let dus altijd op de juiste syntaxis. Voer programma DRAW 12 maar eens in. Binnen twee seconden heeft de MSX een groot aantal steeds maar groter wordende vierkanten op het scherm gezet. In het begin nog netjes op ware grootte maar tenslotte bij S=100 wel 100/4=25 maal vergroot. Dit betekent dat in plaats van 1 beeldelement (pixel) van 4 punten er nu 100 punten worden getekend. Deze vorm van variabeletoewijzing is toepasbaar op alle parameters van DRAW; in programma DRAW 13 bijvoorbeeld wordt met behulp van de instructie A weer een windroos getekend. Zoals u ziet heeft dit programma een regel om A weer in de uitgangstoestand te brengen. Zo vermijden we problemen in volgende programma's. De uitgangstoestand van DRAW is

A0	hoek = 0*90 graden
C15	voorgrondkleur wit
S4	geen vergroting of verkleining
B M 0,0	grafische cursor in de linkerbovenhoek

DRAW-stringvariabelen in DRAW-instructies

Een serie tekeninstructies in DRAW-vorm kan als subroutine fungeren binnen een andere DRAW-instructie. Hoe dat gaat zullen we in de volgende regels laten zien. We gaan uit van het trapje ("R8 U8 R8") als basistekening en noemen die variabele A$. Het vierkant ("R8 U8 L8 D8") heet B$ en het huisje ("R8 U8 L8 E4 F4 G8 F8") C$. Zo'n string is als volgt op te roepen:

X stringvariabelenaam;

Het resultaat van de drie geneste strings ziet u in programma DRAW

14. Dat ziet er weliswaar niet oogverblindend uit maar het geeft wel aan hoe krachtig de methode is en hoe flexibel DRAW-strings aan elkaar geknoopt kunnen worden. We hebben al vermeld dat DRAW-strings net als andere strings (bijvoorbeeld A$=In de sughtende Swaen) mogen worden behandeld. Ook hier geldt de beperking dat een string maximaal 255 tekens lang mag zijn. Bovendien is het verstandig in het geheugen voldoende ruimte te reserveren. Voor vier strings van ieder 200 tekens moet u in ieder geval CLEAR 800 invoeren.

Opmerkingen

- Om richting en grootte van cursorverplaatsingen aan te geven, hoeft u niet binnen de grenzen van het scherm (256*192 punten) te blijven. In feite zijn getallen tussen +32767 en -32768 toegestaan. Hoewel dat geen praktisch nut heeft, zat er misschien bij de ontwerpers van de MSX de wens achter zoveel mogelijk foutmeldingen te voorkomen omdat die er onvermijdelijk toe leiden dat het grafische scherm wordt gewist.
- Er zijn vier geheugenplaatsen gereserveerd waarop u de actuele positie van de cursor kunt aflezen, respectievelijk een nieuwe cursorpositie kunt invoeren (dat laatste komt overeen met de instructie DRAW "B M X Y"). Ook de andere parameters van DRAW kunt u direct in het geheugen aanspreken.

X absoluut	&HFCB3
Y absoluut	&HFBC5
X relatief	&HFBC7
Y relatief	&HFBC9
S (schaalfactor)	&HFCBC
A (draaiing)	&HFCBD
C (voorgrondkleur)	&HF3E9

DRAW 1

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "bm 128,96 u8 r8 u8 r8 d15 l16"
40 GOTO 40
```

DRAW 2

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "bm 128,96 r8 u8 l8 e4 f4 g8 u8 f8"
40 GOTO 40
```

DRAW 3

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "bm 128,96 r8 u8 l8 d8"
40 DRAW "bm 128,96 s80 r8 u8 l8 d8"
50 GOTO 50
```

DRAW 4

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "bm 128,96 r8 u8 r8"
40 GOTO 40
```

DRAW 5

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "bm 128,96 u16 f4 h4 g4"
40 N=N+1
50 ON N GOSUB 70,90,110
60 GOTO 60
70 DRAW "a1"
80 GOTO 30
90 DRAW "a2"
100 GOTO 30
110 DRAW "a3"
120 GOTO 30
```

DRAW 6

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "bm 128,96 n a0 u8 n a1 u8 n a2 u8 n a3 u8"
40 GOTO 40
```

DRAW 7

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "bm 128,96"
40 FOR N=1 TO 4
50 DRAW "bm +0,+8 r4 u4 l4 d4"
60 NEXT N
70 GOTO 70
```

DRAW 8

```
10 COLOR 1,15
20 SCREEN 2
```

```
30 DRAW "s4 a0 bm 50,130"
40 FOR N=1 TO 8
50 DRAW "bm +18,+0 r8 u4 l3 u12 r3 u2 l4 e3
 h3 g3 f3 l4 d2 r3 d12 l3 d4"
60 NEXT N
70 DRAW "s4 a0 bm 50,66"
80 FOR N=1 TO 8
90 DRAW "bm +16,+2 r8 u4 l3 u12 r3 u2 l4 e3
 h3 g3 f3 l4 d2 r3 d12 l3 d4"
100 PAINT STEP(+2,-2)
110 NEXT N
120 GOTO 120
```

DRAW 9

```
10 COLOR 1,4
20 SCREEN 2
30 DRAW "s4 a0"
40 DRAW "bm 128,96 c15 r16 c1 u16 c15 l16 c1 d16"
50 GOTO 50
```

DRAW 10

```
10 COLOR 1,4
20 SCREEN 2
30 DRAW "s4 a0"
40 DRAW "bm 128,96"
50 COLOR 15
60 DRAW "r 16"
70 COLOR 1
80 DRAW "u 16"
90 COLOR 15
100 DRAW "l 16"
110 COLOR 1
120 DRAW "d 16"
130 GOTO 130
```

DRAW 11

```
10 COLOR 1,4
20 SCREEN 2
30 DRAW "bm 128,96 c15 r4 c1 u4 c15 l4 c1 d4"
40 GOTO 40
```

DRAW 12

```
10 COLOR 1,4
20 SCREEN 2
30 FOR N=4 TO 100 STEP 4
40 DRAW "bm 128,96 s=n;r4 u4 l4 d4"
50 NEXT N
60 DRAW "s4"
70 GOTO 70
```

DRAW 13

```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=0 TO 3
```

```
        40 DRAW "bm 128,96 a=n;u8"
        50 NEXT N
        60 DRAW "a0"
        70 GOTO 70

DRAW 14 10 COLOR 1,15
        20 SCREEN 2
        30 A$="s4 r8 u8 r8"
        40 B$="r8 u8 l8 d8"
        50 C$="s8 r8 u8 l8 e4 f4 g8 u8 f8"
        60 DRAW "bm 128,96 x A$;x B$;x C$;"
        70 GOTO 70
```

5.2 PSET

Het programma POS 6 staat in paragraaf 3.7 en PRESET 1 in paragraaf 5.3.

De grafiekinstructie PSET is alleen bruikbaar in SCREEN 2 en 3. In SCREEN 0 en 1 veroorzaakt hij de foutmelding Illegal function call. Het formaat van deze instructie is

```
PSET(X,Y),Z
```

Net als bij de andere grafische instructies kunt u het aan te geven punt (X- en Y-coördinaat) absoluut en relatief adresseren. De eerste methode mag uit het voorgaande bekend zijn. Voor een relatieve adressering is de instructie STEP nodig.

absoluut:

PSET(128,96)	zet een punt op positie 128,96

relatief:

PSET STEP(10,10)	zet een punt, uitgaande van de vorige positie (128,96): 10 punten naar rechts = 138 10 punten naar beneden = 106

Met Z kunt u elk punt afzonderlijk een kleur geven. Als u Z geen waarde geeft, neemt de MSX de actuele voorgrondkleur. PSET kan op dezelfde manier functioneren als de BASIC-instructie PRESET (paragraaf 5.3) maar er is een belangrijk verschil. PSET maakt gebruik van de actuele voorgrondkleur, tenzij anders aangegeven. PRESET daarentegen gebruikt de actuele achtergrondkleur. Meestal

heeft een grafisch scherm grotendeels de achtergrondkleur; PRESET doet dan eigenlijk niets anders dan het aangesproken punt wissen. De instructies zijn dus aan elkaar gelijk als u voor Z de achtergondkleur kiest. Om die kleur niet apart te hoeven invoeren, maken we gebruik van het RAM-adres van deze kleur:

```
PSET(X,Y),PEEK(&HF3EA)=PRESET(X,Y)
```

Door de actuele achtergrondkleur op te slaan als variabele kunnen we de instructie PSET de PRESET-functie te laten vervullen zonder het geheugen uit te lezen. Is dat steeds nodig, dan kunt u de betreffende variabele toevoegen aan PSET. Net als bij DRAW hoeven de waarden voor de coördinaten bij PSET niet binnen de grenzen van het scherm te blijven. De uiterste waarden zijn ook hier -32768 en +32767; overschrijdt u dit bereik, dan reageert de computer met Overflow. Bij elke grafiekinstructie verschijnt de grafische cursor op een bepaalde plaats op het scherm. De waarden die de cursorpositie bepalen, zijn opgeslagen op de vier in paragraaf 5.1 genoemde adressen. &HFCB3 en &HFCB5 geven de actuele absolute positie van de grafische cursor aan. &HFCB7 en &HFCB9 leveren de relatieve positie, bijvoorbeeld wanneer een kruis wordt getekend vanuit een middelpunt of als binnen de instructie PSET gebruik wordt gemaakt van STEP (zie programma POS 6).

Opmerkingen

- Als u alleen de positie van de grafische cursor wilt veranderen (bij de tekstschermen doet u dat met LOCATE), kunt u het beste

  ```
          PRESET(X,Y)
      of
          PSET(X,Y),achtergrondkleur
  ```

  invoeren. Als u bij PSET de achtergrondkleur weglaat, heeft u er direct een ongewenste punt op het scherm bij. Met PRESET krijgt de cursor onzichtbaar een nieuwe plaats (zie programma PRESET 1).
- Het scherm bestaat uit kleurblokjes van 8*1 punten. Binnen zo'n blokje kunt u beter maar 1 kleur gebruiken. Twee kleuren lopen op onvoorspelbare wijze in elkaar over.

5.3 PRESET

De programma's PRESET 1 en 2 staan aan het einde van deze paragraaf.

De instructie PRESET is het omgekeerde van PSET. Met PRESET (point reset) kunnen we punten op een grafisch scherm wissen. 'Wissen' betekent hier dat de kleur van een punt opnieuw wordt gedefinieerd. Er zijn twee mogelijkheden: we kiezen als nieuwe kleur de achtergrondkleur of een nieuwe voorgrondkleur. In het eerste geval wordt de punt onzichtbaar.

PRESET(X,Y)	de te wissen punt krijgt de achtergrondkleur
PRESET(X,Y),Z	de te wissen punt krijgt kleur nummer Z

Programma PRESET 1 demonstreert het 'vlakgom-effect' van deze instructie. In regel 30-60 wordt met zwart (kleur 1) een rechthoekig vlak getekend, dat vervolgens in regel 70-110 met de achtergrondkleur (wit = kleur 15) wordt weggevaagd. PSET met de kleuraanduiding 'achtergrond' heeft hetzelfde resultaat als PRESET zonder kleurtoekenning. Regel 90 zou dus ook kunnen luiden

```
90 PSET(M,N),15
```

en omgekeerd regel 50 ook

```
50 PRESET(M,N),1
```

Het afwisselend gebruik van PSET en PRESET heeft het voordeel dat u niet steeds de voorgrondkleur respectievelijk de achtergrondkleur hoeft aan te geven. Als u niet steeds wilt wisselen van instructie, kunt u de adressen van de voorgrond- en de achtergrondkleur ook direct aanspreken. Dat zijn

&HF3E9	voorgrondkleur
&HF3EA	achtergrondkleur

Daarmee beschikken we over de volgende equivalenten:

```
PSET(X,Y)   = PRESET(X,Y),PEEK(&HF3E9)

PRESET(X,Y) = PSET(X,Y),PEEK(&HF3EA)
```

Ook bij deze instructies kunt u weer absoluut of relatief adresseren. In het laatste geval moet u ter onderscheiding het woord STEP toevoegen.

```
PRESET STEP(10,-10)
```

Uitgaande van de actuele cursorpositie krijgt het punt 10 plaatsen naar rechts en 10 plaatsen naar boven nu de achtergrondkleur. Bij gebruik van PSET zou het de voorgrondkleur zijn geworden. De coördinaten mogen waarden tussen -32768 en +32768 hebben. Overschrijdt u dit bereik, dan volgt de foutmelding Overflow error. De kleuren mogen de codes 0-15 hebben. Kleur 0 betekent 'doorzichtig'; dat heeft dus geen effect op al aanwezige kleuren. Kleurwaarden buiten dit bereik veroorzaken de foutmelding Illegal function call. Elke keer als PRESET wordt gebruikt, verandert de actuele positie van de grafische cursor. Een voorbeeld:

```
PRESET(128,96)
CIRCLE STEP(0,0),30
```

De cursor krijgt een nieuwe plaats die niet zichtbaar is; het middelpunt is niet verschoven ten opzichte van die plaats. Door de onzichtbare verplaatsing van de grafische cursor (net als met DRAW"B M X,Y") kunt u PRESET gebruiken om de beginpositie van tekstuitvoer op een grafisch scherm aan te geven (zie programma PRESET 2). Zo vervult de instructie PRESET dezelfde functie als LOCATE in een tekstmodus.

PRESET 1

```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=100 TO 120
40 FOR M=80 TO 100
50 PSET(M,N)
60 NEXT M,N
70 FOR N=100 TO 120
80 FOR M=80 TO 100
90 PRESET(M,N)
100 NEXT M
110 NEXT N
120 GOTO 120
```

PRESET 2

```
10 COLOR 1,15
20 SCREEN 2
30 OPEN "grp:" FOR OUTPUT AS #1
40 PRESET(128,96)
50 PRINT #1,"MSX";
60 CLOSE
70 GOTO 70
```

5.4 POINT

De programma's POINT 1-3 staan aan het einde van deze paragraaf.

Het grafische scherm met hoog oplossend vermogen (SCREEN 2) heeft een coördinatensysteem met 256*192 punten; het grovere SCREEN 3 heeft 64*48 punten. In SCREEN 3 kan ieder beeldelement (een blokje van 4*4) precies 1 kleur bevatten. In SCREEN 2 is het een beetje ingewikkelder. In theorie zouden bijna 50.000 beeldpunten afzonderlijk kunnen worden aangesproken, maar voor de kleurtoekenning gaat die vlieger niet op. Verticaal kunnen de punten allemaal verschillende kleuren krijgen. Horizontaal hebben we te maken met groepjes van 8 punten waarin steeds maximaal twee kleuren mogen voorkomen (punt 0-7 twee kleuren, punt 8-15 twee kleuren, enzovoort). Proberen we een derde kleur toe te voegen, dan wordt die de overheersende voor- of achtergondkleur en dat werkt ook door op andere al aanwezige kleuren. Met de instructie POINT kunt u nagaan welke kleur elk punt op het scherm heeft. Als u de coördinaten van het gewenste punt invoert, verschijnt er een getal op het scherm dat u in de kleurentabel kunt nazoeken. Als voorbeeld nemen we programma POINT 1, waarin een vlak van 20*20 helemaal zwart wordt gekleurd (kleur 1). Alle punten binnen dit vlak, inclusief het kader, hebben dus die kleurwaarde. In het volgende voorbeeld, POINT 2, kennen we de kleurinhoud van een deel van het scherm toe aan een array (A) die vervolgens weer op het scherm komt in de vorm van kleurcodes. In regel 50 wordt op het scherm een vierkantje van 8*8 getekend in kleur 2, tegen een achtergrond van kleur 3. In de volgende regels wordt de informatie punt voor punt ingelezen in array A. Vervolgens worden de waarden van A keurig achter elkaar op het tekstscherm weergegeven en zien we het vierkantje weer terug, maar nu in de vorm van getallen. Met een paar kleine wijzigingen in het bovenstaande programma kunt u ook de vorm van letters en tekens punt voor punt van het scherm lezen en vergroot weergeven op een tekstscherm. Laten we dat eens met de letter A proberen (zie programma POINT 3). De A is opgebouwd uit zwarte punten, dus komt hij in de vorm van enen (zwart = 1) op het tekstscherm terug. In regel 70 kunt u zelf karakters invoeren. Met wat fantasie kunt u zich misschien de moeite besparen de karaktergenerator (de listing daarvan staat in paragraaf 9.14) in te typen omdat u met dit programma iets dergelijks kunt bereiken. Behalve voor de genoemde geintjes is de instructie POINT ook bruikbaar bij het grotere werk. Als u een scherm met hoog oplossende grafiek wilt uitvoeren op een matrixprinter moet het scherm punt voor punt worden gelezen. Dat gaat uitstekend met de instructie POINT. Het spel FREDDIE in paragraaf 9.15 laat dat zien (bijvoorbeeld de regels 1720 en 1730). Denk er bij het schrijven van een hardcopy-routine aan dat alleen bepaalde kleuren op papier moeten komen. Meestal wordt een kleurenplaatje zwart-wit geprint. Stel dat kleur 3 de

belangrijkste informatie bevat dan voert u in:

```
IF POINT(X,Y)=3 THEN LPRINT ...
```

In een spel is het soms nodig informatie op het scherm in te voeren in een programma. Denk eens aan een mannetje dat over een rode muur loopt en over gaten moet springen. Omdat die gaten natuurlijk niet rood zijn, kunnen we met POINT controleren of het mannetje op de goede plaats zijn voeten neerzet (de kleur onder zijn voeten is dan rood).

Opmerkingen

- De instructie POINT komt erop neer dat een stuk van het geheugen dat niet in gebruik is in de tekstmodi wordt gelezen. POINT is dus op SCREEN 0 en 1 niet van toepassing.
- U mag POINT ook toepassen op punten buiten het scherm als de coördinaten maar binnen de grenzen -32768 tot +32768 liggen. Het resultaat is dan de waarde -1. Buiten die grenzen krijgt u de melding Overflow error.
- Eventueel kunt u in plaats van de instructie POINT te gebruiken met VPEEK rechtstreeks waarden in het video-RAM lezen. Daarvoor is gedegen kennis van de indeling van het VRAM noodzakelijk. Kijk daarvoor bij SCREEN, VDP en BASE. Bovendien heeft u dan een behoorlijk ingewikkeld geheugendecoderingsprogramma nodig. Dat geldt in ieder geval als het om SCREEN 2 gaat omdat daar de kleuren worden bepaald door gegevens uit de colour table en de pattern name table samen.
- Er zouden problemen kunnen ontstaan doordat de kleur van een vlak niet aan de regels voldoet in combinatie met de kleur van de rand van het vlak. Een test op de kleur van de rand zou dus handig zijn. Met POINT kunt u de waarde van de randkleur testen en vervolgens zonder problemen met PAINT het vlak inkleuren.

POINT 1

```
10 COLOR 1,15
20 SCREEN 2
30 LINE(128,96)-(148,76),1,BF
40 A=POINT(138,85)
50 FOR N=1 TO 1000
60 NEXT N
70 SCREEN 0
80 PRINT A
```

POINT 2

```
10 COLOR 1,3
20 SCREEN 2
30 WIDTH 37
40 DIM A(12,12)
50 LINE(128,96)-(135,103),2,BF
60 FOR N=127 TO 136
70 FOR M=95 TO 106
80 A(N-126,M-94)=POINT(N,M)
90 NEXT M
100 NEXT N
110 SCREEN 0
120 FOR N=1 TO 10
130 FOR M=1 TO 10
140 PRINT A(M,N);
150 NEXT M
151 PRINT
160 NEXT N
```

POINT 3

```
10 COLOR 1,3
20 SCREEN 2
30 WIDTH 37
40 DIM A(10,10)
50 OPEN "grp:" FOR OUTPUT AS #1
60 DRAW "bm 127,97"
70 PRINT #1, "A"
80 FOR N=127 TO 136
90 FOR M=97 TO 106
100 A(N-126,M-96)=POINT (N,M)
110 NEXT M
120 NEXT N
130 SCREEN 0
140 FOR N=1 TO 10
150 FOR M=1 TO 10
160 PRINT A(M,N);
170 NEXT M
180 PRINT
190 NEXT N
200 CLOSE
```

5.5 LINE

De programma's LINE 1-13 staan aan het einde van deze paragraaf en DRAW 6 staat in paragraaf 5.1.

Met de instructie LINE zijn drie dingen mogelijk: lijnen trekken, rechthoeken tekenen en getekende rechthoeken inkleuren.

Lijnen

U kunt ofwel het begin- en eindpunt van een lijn aangeven ofwel uitgaan van de actuele cursor en alleen het eindpunt opgeven. In het laatste geval moet u wel opletten als u de computer net hebt aangezet omdat de cursor zich dan op positie 0,0 bevindt. Programma LINE 1 laat beide mogelijkheden zien. De instructie in regel 30 geeft aan dat vanaf het midden van het scherm (128,96) een lijn moet worden getrokken naar (150,150). In regel 40 staat geen beginpunt zodat de MSX eenvoudig van de actuele positie naar het genoemde punt (160,120) gaat. Veranderen we regel 30 in

```
30 LINE-(150,150)
```

en starten we het programma opnieuw, dan verschijnt er een nieuwe lijn op het scherm. Die begint op de actuele cursorpositie (160,120) en loopt naar het in regel 30 aangegeven punt (150,150). Om toch weer op de uitgangspositie terug te komen (0,0) moet u een regel invoegen.

```
25 DRAW "B M 0,0"
```

Nu trekt het programma een lijn van (0,0) naar (150,150). Het is dus heel belangrijk eerst goed te controleren waar de cursor zich bevindt om te voorkomen dat nog te trekken lijnen over elkaar gaan lopen. Behalve de hierboven toegepaste absolute adressering kunt u bij de instructie LINE ook relatief adresseren met STEP (zie programma LINE 2).

regel 40 — Vanaf het midden van het scherm (128,96) wordt een lijn getrokken naar (150,150). De adressering is absoluut.

regel 50 — Vanuit de actuele positie (150,150) gaat de cursor (zonder lijn) met STEP 10 punten naar rechts en 10 punten naar onder. Daarna gaat de lijn, geadresseerd zonder STEP en dus absoluut naar positie 180,180.

regel 60 — Hier hebben zowel het beginpunt als het eindpunt van de lijn de aanduiding STEP zodat het in beide geval-

len om relatieve adressen gaat.
Het beginpunt is (180,180) + (-20,-20) = (160,160);
het eindpunt wordt (160,160) + (-40,-40) = (120,120).

Het is niet in overeenstemming met wiskundige afspraken om de oorsprong van de coördinaten (0,0) in de linkerbovenhoek van het scherm te plaatsen. Wie daaraan niet gewend is, kan in verwarring raken door de manier waarop de verschillende richtingen worden aangegeven. We zetten daarom de mogelijkheden bij relatieve adressering even op een rijtje.

LINE STEP(x1,y1)-(x2,y2)

x1	y1	cursor verder naar
positief		rechts
negatief		links
	positief	onder
	negatief	boven

Volgens de wiskundige afspraken zou (0,0) linksonder moeten liggen en het verste punt (255,191) rechtsboven, maar het is helaas niet anders en u zult het met de tabel hierboven moeten doen. Ten opzichte van de instructie DRAW heeft LINE het voordeel dat er niet de beperking is van 8 tekenrichtingen (zie programma DRAW 6 in paragraaf 5.1). Dit is gesneden koek als u die paragraaf al heeft doorgewerkt. Met LINE kunt u alle kanten op. Zo tekent het volgende programma, LINE 3, een heel vlak vol. Hierin worden vanuit een vast punt (128,96) lijnen getekend naar alle punten op een lijn (N,0), hetgeen resulteert in een opgevulde driehoek. In programma LINE 4 is de straal eerst groot en wordt geleidelijk aan steeds kleiner. Zodra de afstand tussen de eindpunten 0 is geworden (M=1), wordt de stralenbundel net zo dicht als in LINE 3.

Rechthoeken

LINE biedt de mogelijkheid om met de eindpunten van lijnen de omtrek van een rechthoek te tekenen. Met de kennis die u tot nu toe heeft vergaard, moet u daarbij de absolute adressering gebruiken, zoals in programma LINE 5 is gebeurd. Maar als u goed kijkt naar de coördinaten van de hoekpunten zijn er eigenlijk maar twee X- en twee Y-waarden nodig.

x1 = 128   x2 = 148
y1 = 96   y2 = 76

Bent u ook benieuwd welke lijn er verschijnt met deze waarden in

een LINE-instructie? Run dan programma LINE 6 maar eens. Het vierkant kent u al; de diagonaal van linksonder naar rechtsboven is nieuw. De MSX heeft er geen moeite mee om uit die twee punten de andere twee hoekpunten te bepalen.

bekende coördinaten	(128,96)	(148,76)
nieuwe coördinaten	(128,76)	(148,96)

De computer zet ze dan ook nog in een volgorde waarin het vierkant op een logische manier wordt getekend.

1) 128,96 2) 128,76 3) 148,76 4) 148,96

of

1) 128,96 2) 148,96 3) 148,76 4) 128,76

Door een verkeerde volgorde zou een figuur ontstaan met voor een deel goede zijden en een of twee diagonalen. Met de B (van box) kan de LINE-instructie worden uitgebreid om een rechthoek te tekenen.

```
LINE(x1,y1)-(x2,y2),,B
```

of, als het beginpunt al bekend is

```
LINE-(x2,y2),,B
```

Over het waarom van de twee komma's voor de B later meer. Programma LINE 7 geeft een idee van de snelheid waarmee rechthoeken worden getekend als we bovenstaande instructie gebruiken. Snel genoeg naar uw zin? In een oogwenk staan er 1000 rechthoeken op het scherm. U begrijpt dat ook voor rechthoeken de mogelijkheid bestaat de coördinaten relatief aan te geven met behulp van de instructie STEP. Dat is al besproken, maar we geven hier nog een voorbeeld: LINE 8. Het is nu wel duidelijk dat u om een rechthoek te tekenen alleen de diagonaal van linksboven (0,0) naar rechtsonder (10,10) hoeft aan te geven. Alle 10 rechthoeken in dit voorbeeld liggen op een diagonaal omdat het in LINE 8 om vierkanten gaat en het eerste punt van een nieuw vierkant met STEP(0,0) aansluit op het vorige. In het volgende voorbeeld is alleen de doelpositie (X2,Y2) van LINE gegeven. De eerste positie van het nieuwe vierkant is gelijk aan de tweede positie van het vorige (zie programma LINE 9). Maak nu gewapend met de kennis die u in bovenstaande programma's hebt opgedaan in plaats van een scherm met hoog oplossend vermogen een gerasterd beeld dat helemaal uit kleine rechthoekjes bestaat.

Gekleurde vulling

Aan de letter B van een LINE-instructie mag nog een F worden toegevoegd. Die F (van fill) geeft aan dat de rechthoek moet worden opgevuld met de kleur van de omtreklijnen. Bekijk dat eens met programma LINE 10. In het begin ziet het er leuk uit maar na korte tijd lijkt er niets meer te veranderen. Er gebeurt wel van alles maar dat is niet zichtbaar meer omdat de nieuwe vlakken voor het grootste deel stukken overlappen waar al wat staat. Er zijn twee manieren om het laatste stuk van dit kunstwerk toch te zien te krijgen.
Met kleinere rechthoeken is het scherm niet zo vlug vol. Neem dus in plaats van willekeurig (grote) rechthoeken kleintjes van 5*5 door de volgende regels te veranderen.

```
50 X2=X1+5
70 Y2=Y1+5
```

Als u wilt, kunt u nu helemaal tot 1000 meetellen. Een andere manier om het bovenstaande te bereiken is met STEP het tweede punt (X2,Y2) uit het eerste (X1,Y1) te berekenen.

```
80 LINE(x1,y1)-STEP(+5,+5),,BF
```

De zojuist veranderde regels 50 en 70 mogen nu helemaal weg.

Neem voor elk vierkant een andere kleur om ze beter van elkaar te onderscheiden. Verander programma LINE 10 of neem LINE 11. In regel 80 wordt met een toevalsfactor (RND) een kleur gekozen voor de rechthoek. De veelzijdigheid van MSX-BASIC wordt hier weer eens bewezen, want we kunnen regel 80 ook weglaten en de RND-functie als parameter aan de instructie LINE toevoegen.

```
90 LINE(X1,Y1)-(X2,Y2),INT(RND(1)*15),BF
```

U ziet nu ook wat twee komma's aan het eind van de LINE-instructie te betekenen hebben. Als er niets staat geldt de voorgrondkleur. Het valt direct op dat er op veel plaatsen van het scherm kleuren door elkaar lopen en u weet ook waarom. Het oplossend vermogen is bij gebruik van kleuren helaas een stuk kleiner dan in een monochrome modus. De kleur die het laatst op het scherm is verschenen, heeft de overhand en dat vaak over een hele serie punten van een beeldlijn. Met wat denkwerk en voorzichtigheid kunt u de vrije ruimte van het VRAM (16 Kbyte) gebruiken om de kleuren beter uit elkaar te houden. Dan kunt u echt met 16 kleuren in SCREEN 2 schilderen. Tot slot nog twee programma's om wat meer te weten te komen over het door elkaar lopen van de kleuren. Eerst LINE 12. Het resultaat van dit programma is niet zo fraai want de verschillende kleuren liggen te dicht bij

elkaar. In het volgende programma zijn de stappen groter en ontstaan er minder rechthoeken, dus een veel kleinere kans op overlapping (zie programma LINE 13). Hiermee wil niet gezegd zijn dat kleuren altijd zo ver uit elkaar moeten liggen. Meer details over de opbouw van het scherm in de hi-res modus vindt u in paragraaf 1.1 over SCREEN.

LINE 1
```
10 COLOR 1,15
20 SCREEN 2
30 LINE(128,96)-(150,150)
40 LINE-(160,120)
50 GOTO 50
```

LINE 2
```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "bm 0,0"
40 LINE(128,96)-(150,150)
50 LINE STEP(10,10)-(180,180)
60 LINE STEP(-20,-20)-STEP(-40,-40)
70 GOTO 70
```

LINE 3
```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=10 TO 240
40 LINE(128,96)-(N,0)
50 NEXT N
60 GOTO 60
```

LINE 4
```
10 COLOR 1,15
20 SCREEN 2
30 M=241
40 FOR N=10 TO 240 STEP M
50 LINE(128,96)-(N,0)
60 NEXT N
70 M=M-10
80 IF M<1 THEN GOTO 110
90 CLS
100 GOTO 40
110 GOTO 110
```

LINE 5
```
10 COLOR 1,15
20 SCREEN 2
30 LINE(128,96)-(148,96)
40 LINE-(148,76)
50 LINE-(128,76)
60 LINE-(128,96)
70 GOTO 70
```

LINE 6
```
10 COLOR 1,15
20 SCREEN 2
30 LINE(128,96)-(148,76)
40 GOTO 40
```

LINE 7
```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=1 TO 1000
40 X1=INT(RND(1)*255)
50 X2=INT(RND(1)*255)
60 Y1=INT(RND(1)*191)
70 Y2=INT(RND(1)*191)
80 LINE(X1,Y1)-(X2,Y2),,B
90 NEXT N
100 GOTO 100
```

LINE 8
```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "bm 128,96"
40 FOR N=1 TO 10
50 LINE STEP (0,0)-STEP(10,10),,B
60 NEXT N
70 GOTO 70
```

LINE 9
```
10 COLOR 1,15
20 SCREEN 2
30 DRAW"bm 128,96"
40 FOR N=1 TO 10
50 LINE-(128+(N*10),96+(N*10)),,B
60 NEXT N
70 GOTO 70
```

LINE 10
```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=1 TO 1000
40 X1=INT(RND(1)*255)
50 X2=INT(RND(1)*255)
60 Y1=INT(RND(1)*191)
70 Y2=INT(RND(1)*191)
80 LINE(X1,Y1)-(X2,Y2),,BF
90 NEXT N
100 GOTO 100
```

LINE 11
```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=1 TO 1000
40 X1=INT(RND(1)*255)
50 X2=INT(RND(1)*255)
60 Y1=INT(RND(1)*191)
```

```
70 Y2=INT(RND(1)*191)
80 COLOR INT(RND(1)*15)
90 LINE(X1,Y1)-(X2,Y2),,BF
100 NEXT N
110 GOTO 110
```

LINE 12

```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=100 TO 5 STEP-1
40 LINE(128,96)-STEP(+N,+N),INT(RND(1)*15),BF
50 NEXT N
60 GOTO 60
```

LINE 13

```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=100 TO 5 STEP-8
40 LINE(128,96)-STEP(+N,+N),INT(RND(1)*15),BF
50 NEXT N
60 GOTO 60
```

5.6 CIRCLE

De programma's CIRCLE 1-11 staan aan het einde van deze paragraaf.

CIRCLE is een van de krachtigste instructies van MSX-BASIC. Toch zijn al veel mensen tot wanhoop gedreven omdat het PAL-systeem elke cirkel op een TV-toestel vervormt tot een ellips (zie programma CIRCLE 1). We zullen niet ingaan op de oorzaak van dit verschijnsel, maar het direct corrigeren. Daartoe veranderen we regel 30 in CIRCLE 1.

```
30 CIRCLE(128,96),50,,,,4/3
```

Als het nu nog niet in orde is, heeft u waarschijnlijk zelf het toestel verkeerd afgesteld. Maak dat eerst in orde (bijvoorbeeld met het testbeeld). De instructie CIRCLE is alleen bestemd voor de grafische schermen 2 en 3. In de andere modi volgt de foutmelding Illegal function call. CIRCLE heeft zes parameters:

```
CIRCLE(A,B),C,D,(E,F),G
```

A en B

A en B geven samen het middelpunt van een cirkel aan. De

adressering kan absoluut zijn of relatief (met STEP). De programma's CIRCLE 2 en CIRCLE 3 geven van beide manieren een voorbeeld. Bij de relatieve adressering worden de cirkelpunten berekend vanuit het middelpunt; in CIRCLE 3 is dat positie 128,96.

C

Parameter C geeft de straal van de cirkel aan. C moet liggen in het bereik van de gehele getallen -32768 tot +32768. Daarbuiten volgt de foutmelding Overflow error. De cirkel hoeft dus niet op het scherm te passen.

D

Dit is de parameter voor de kleur van de cirkel. Bij de MSX hebt u het niet helemaal voor het kiezen. Als u geen rekening houdt met het lagere oplossend vermogen van een scherm met kleur, kunnen de kleuren door elkaar gaan lopen (zie programma CIRCLE 4). Een wat grotere afstand tussen de cirkels kan dit euvel voorkomen.

```
30 FOR N=1 TO 200 STEP 8
```

E en F

E en F dienen om begin- en eindpunt van een (deel van een) cirkel aan te geven. Het lastige is dat de waarden in radialen moeten worden aangegeven in plaats van graden. Bovendien moet u er aan wennen dat de cirkel begint op 90 graden (drie uur) en tegen de wijzers van de klok in loopt. De waarden moeten liggen tussen +2*pi (+6.21318) en -2*pi (-6.21318). Buiten dit bereik volgt de foutmelding Illegal function call. We tekenen eerst een halve cirkel, beginnend op 90 graden. Een cirkel (360 graden) omvat 2*pi radialen; een halve cirkel (180 graden) dus pi radialen (3.14159): zie programma CIRCLE 5. Vervolgens krijgen we een kwart cirkel, beginnend op 180 graden (programma CIRCLE 6). CIRCLE 7 tekent een gearceerde cirkel. Als u zich nog de bespreking van DRAW en MGL (macro graphics language) herinnert (paragraaf 5.1), zult u nu wel begrijpen hoe we een eenvoudig schijfdiagram kunnen maken (programma CIRCLE 8). CIRCLE 9 is een mooi voorbeeld van de prachtige vormen die u met deze instructie op het scherm kunt toveren. De rekencapaciteit van de computer kunnen we mooi gebruiken om graden om te rekenen in radialen. Dat doen we door het aantal graden te delen door 57.2959 (die waarde is het quotient van 360 graden gedeeld door 2 pi).

graden	radialen
360	6.38218 (2 pi)
270	4.71238 (1.5 pi)
180	3.14159 (pi)
60	1.047197 (pi:3)

Omdat het vervelend is die waarden steeds op te zoeken of uit te rekenen, maken we er een functie van.

```
10 DEF FNA(GRAD)=GRAD/57.2958
20 INPUT "hoeveel graden?";GRAD
30 PRINT FNA(GRAD)
```

G

Deze parameter zijn we aan het begin van deze paragraaf al tegengekomen. Hij dient om cirkels om te vormen tot ellipsen of omgekeerd. De waarden liggen tussen 1/260 en 260. Bij de uiterste waarden blijven er van de cirkels lijnen over, respectievelijk in de X- en in de Y-richting. Dat is eigenlijk niet de bedoeling van deze instructie en daarom worden meestal waarden tussen 0 en 2 gekozen. Als G gelijk is aan 1, treedt er geen vervorming op; dit is de uitgangswaarde. Bij een waarde groter dan 1 wordt de cirkel in de Y-richting uitgerekt, bij een waarde kleiner dan 1 in de X-richting. Programma CIRCLE 10 demonstreert dit effect. Uit de voorbeelden is wel duidelijk geworden dat u niet per se alle parameters hoeft te gebruiken. Het middelpunt (A,B) en de straal C moeten natuurlijk wel bekend zijn. De kleur is de actuele voorgrondkleur. Dit is een van de parameters die u over kunt slaan door een komma te gebruiken. In programma CIRCLE 11 voeren we middelpunt, straal en puntafstand in. Met behulp van enkele BASIC-instructies berekent het programma de benodigde waarden en tekent dan met PSET de cirkel. Daarnaast wordt ook een cirkel getekend met de instructie CIRCLE. Dat gaat veel sneller omdat het in machinetaal gebeurt. Het voordeel van het BASIC-programma is dat u met de extra parameter P de afstand tussen de punten van de cirkel kunt bepalen.

CIRCLE 1
```
10 COLOR 1,15
20 SCREEN 2
30 CIRCLE(128,96),50
40 GOTO 40
```

CIRCLE 2
```
10 COLOR 1,15
20 SCREEN 2
30 CIRCLE(128,96),20
40 CIRCLE(64,80),20
50 GOTO 50
```

CIRCLE 3
```
10 COLOR 1,15
20 SCREEN 2
30 PSET(128,96)
40 CIRCLE STEP(-20,0),30
50 CIRCLE STEP(60,30),30
60 GOTO 60
```

CIRCLE 4
```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=1 TO 100
40 CIRCLE(128,96),N,INT(RND(1)*15)
50 NEXT N
60 GOTO 60
```

CIRCLE 5
```
10 COLOR 1,15
20 SCREEN 2
30 CIRCLE(128,96),30,,0,3.14159
40 GOTO 40
```

CIRCLE 6
```
10 COLOR 1,15
20 SCREEN 2
30 CIRCLE(128,96),30,,3.14159,4.712385#
40 GOTO 40
```

CIRCLE 7
```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=0 TO 6.21318 STEP 1.55795
40 CIRCLE(128,96),30,,N,N+1
50 NEXT N
60 GOTO 60
```

CIRCLE 8
```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=0 TO 4.712385# STEP 1.55995
40 CIRCLE(128,96),30,,N,N+1.55795
50 Z=Z+1
60 IF Z=1 THEN DRAW "n u30"
70 IF Z=2 THEN DRAW "n l30"
```

```
80 IF Z=3 THEN DRAW "n d30"
90 IF Z=4 THEN DRAW "n r30"
100 FOR M=1 TO 100:NEXT M
110 NEXT N
120 GOTO 120
```

CIRCLE 9

```
10 COLOR 1,15
20 SCREEN 2
30 Z=1
40 FOR N=0 TO 6.28318 STEP .1
50 Z=Z+1
60 CIRCLE(128,96),Z,,0,N
70 NEXT N
80 GOTO 80
```

CIRCLE 10

```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=2 TO 1/260 STEP -.1
40 CIRCLE(128,96),60,,,,N
50 NEXT N
60 GOTO 60
```

CIRCLE 11

```
10 COLOR 15,4
20 CLS
30 INPUT "Radius, X, Y, puntafstand (R,X,Y,P) ";R,X,Y,P
40 SCREEN 2
50 A=57.2985
60 CIRCLE(X,Y),R-10
70 FOR B=1 TO 360 STEP P
80 X1=R*COS(B/A)+X
90 Y1=R*SIN(B/A)+Y
100 PSET(X1,Y1),1
110 NEXT B
120 GOTO 120
```

5.7 PAINT

De programma's PAINT 1-4 staan aan het einde van deze paragraaf.

MSX-BASIC beschikt, evenals andere BASIC-versies, over grafische instructies om punten te zetten (PSET) en lijnen te trekken (LINE). Daarnaast kunnen we vlakken ook nog anders weergeven: met LINE ,,B geven we de omtrek aan en met de speciale instructie PAINT kleuren we het vlak in. Het eerste programma, PAINT 1, toont verschillende manieren om vlakken te tekenen.

regel 30	een cirkel (met CIRCLE)
regel 40	een rechthoek (met LINE ,,B)
regel 50	een rechthoek (met DRAW ...R U L D)
regel 60-90	een rechthoek (met LINE, vier maal herhaald)

Het gaat ons hier niet om de omtrek, maar om de inhoud van de vlakken. De instructie PAINT werkt vanaf een punt binnen de omtrek van de figuur. Bij een cirkel is de keuze niet moeilijk: daarvoor kunnen we gewoon het middelpunt (128,96) nemen. Voorlopig gebruiken we als PAINT-kleur dezelfde kleur als de omtrek van de figuren. Voor de vier figuren kiezen we de volgende coördinaten:

PAINT(128,96)	cirkel
PAINT(85,105)	rechthoek met LINE ,,B
PAINT(75,115)	rechthoek met DRAW
PAINT(65,125)	rechthoek met LINE (4 x)

(Zie programma PAINT 1a). Evenals bij de andere grafische instructies hoeft bij PAINT het punt waar het inkleuren begint niet absoluut te worden geadresseerd. Met een relatief adres rekent de computer het startpunt zelf uit. Bij een cirkel is de actuele cursorpositie het middelpunt (128,96). Als dat het startpunt voor PAINT moet worden, voert u in

```
35 PAINT STEP (0,0)
```

Hiermee geeft u aan dat de actuele cursorpositie (het middelpunt van de cirkel) tevens het startpunt is voor PAINT. Zoals gezegd is de kleur bij deze instructie dezelfde als de kleur van de omtrek. U zou op het idee kunnen komen de kleur te veranderen met een COLOR-instructie voor PAINT. Het resultaat is dan niet helemaal wat u ervan verwacht had, want met bijvoorbeeld

```
33 COLOR 3
```

krijgt het hele scherm een nieuwe kleur en daarmee verdwijnt wat er stond. De computer gaat namelijk net zolang door met inkleuren

tot de cursor een punt met dezelfde kleur (van de omtrek) tegenkomt. In ons voorbeeld kon dat niet omdat kleur 3 nog niet op het scherm aanwezig was. Wilt u vlakken met verschillende kleuren, dan moet u de randkleuren aanpassen. Programma PAINT 2 tekent een aantal cirkels met verschillende kleuren. Helaas treden ook hierbij kleuroverlappingen op. U moet dus voorzichtig omgaan met veel verschillende kleuren dicht op elkaar. De MSX kan nu eenmaal maar twee kleuren tegelijk in een vlakje van 8*1 verwerken. De COLOR-instructie is overbodig omdat we de kleurinformatie zowel in CIRCLE als in PAINT kunnen opnemen (zie ook CIRCLE in paragraaf 5.6 en andere voorbeelden in deze paragraaf). PAINT heeft twee parameters. Met de eerste bepaalt u de kleur waarmee de figuur wordt gevuld. De wijziging van programma PAINT 2 (PAINT 2a) heeft hierop betrekking. De COLOR-instructie wordt weggelaten. Dat is maar goed ook omdat die doorwerkt in SCREEN 0 en 1. De tweede parameter regelt de kleur van de rand. Dat kan alleen in SCREEN 3.

```
PAINT (X,Y), kleur van vlak, kleur van rand
```

Als voorbeeld een programma (PAINT 3) dat drie concentrische cirkels tekent en de binnenste cirkel vult met de kleur van de buitenste cirkel. De kleurvermenging moet u maar voor lief nemen.

Opmerkingen

- U kent nu twee instructies om op een grafisch scherm een vlak te vullen met een kleur. De instructie LINE,BF (box fill) is alleen geschikt voor rechthoekige vlakken, PAINT daarentegen voor elke denkbare vorm. Vooral bij grotere vlakken is LINE,BF aan te bevelen, omdat de snelheid dan gaat tellen.

```
10 SCREEN 2               10 SCREEN 2
20 PAINT(128,96),1        20 LINE(0,0)-(255,191),1,BF
30 GOTO 30                30 GOTO 30

15 seconden               1 seconde
```

- De instructie PAINT werkt alleen de vier hoofdrichtingen af. Diagonale randen kunnen zo gemakkelijk aan de aandacht ontsnappen met als gevolg dat daarna het hele scherm met de PAINT-kleur wordt gevuld. Het volgende BASIC-programma laat zien hoe de instructie PAINT in zijn werk gaat (PAINT 4). MET POINT(N,M) wordt voor ieder punt van de figuur bekeken of de kleur gelijk is aan de randkleur. Dit programma is nog langzamer dan LINE,BF in het voorbeeld hierboven.

Wie met PEEK en POKE wil werken, vindt het werkgeheugen voor de

instructie PAINT vanaf adres &HF949. Adres &HFCB2 bevat de kleur van de rand die bij PAINT is toegestaan.

PAINT 1
```
10 COLOR 1,15
20 SCREEN 2
30 CIRCLE(128,96),10
40 LINE(80,110)-(150,70),,B
50 DRAW "bm 70,120 r90 u60 l90 d60"
60 LINE(60,130)-(170,130)
70 LINE -(170,50)
80 LINE -(60,50)
90 LINE -(60,130)
100 GOTO 100
```

PAINT 1a
```
35 PAINT(128,96)
45 PAINT(85,105)
55 PAINT(75,115)
95 PAINT(65,125)
```

PAINT 2
```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=1 TO 14
40 COLOR N
50 CIRCLE(128,96),50
60 PAINT STEP(0,0)
70 NEXT N
80 GOTO 80
```

PAINT 2a
```
40 REM
50 CIRCLE(128,96),50,N
60 PAINT STEP(0,0),N
```

PAINT 3
```
10 COLOR 1,15
20 SCREEN 3
30 FOR N=1 TO 3
40 CIRCLE(128,96),N*10,N
50 NEXT N
60 PAINT(128,96),3,1
70 GOTO 70
```

PAINT 4
```
10 COLOR 1,15
20 SCREEN 2
30 LINE(128,96)-(158,46),,B
40 FOR N=129 TO 157
50 FOR M=95 TO 47 STEP-1
60 IF POINT(N,M)<>1 THEN PSET(N,M)
70 NEXT M
80 NEXT N
90 GOTO 90
```

5.8 SPRITE$(N)

De programma's SPRITE 1-4 staan aan het eind van deze paragraaf.

De meeste grafiekinstructies zijn alleen bestemd voor de grafische schermen. In de andere schermmodi leiden ze tot de foutmelding Illegal function call. SPRITE$(N) is een grafische instructie die ook in SCREEN 1 bruikbaar is omdat daarin sprites kunnen worden weergegeven. In SCREEN 0 kan dat niet en toch volgt daar geen foutmelding. Het voordeel van SPRITE$(N) is dat u daarmee beschikt over een gedimensioneerde variabeletoewijzing die geen geheugenruimte inneemt in het RAM maar in het VRAM. In SCREEN 0 kunt u dan series tekens van maximaal 32 stuks opslaan zonder eerst met CLEAR geheugenruimte te hoeven reserveren of met DIM een variabele te dimensioneren. Met de tweede parameter van SCREEN geeft u aan hoe u de ruimte voor SPRITE$-variabelen wilt verdelen.

SCREEN ,0 en SCREEN ,1	256 variabelen van maximaal 8 tekens
SCREEN ,2 en SCREEN ,3	64 variabelen van maximaal 32 tekens

Deze toepassing van de instructie SPRITE$(N) is ook in SCREEN 1-3 mogelijk. In feite gaat het om variabeletoewijzing; daarom kunt u zelf bepalen hoe u de betreffende geheugenruimte verdeelt over echte sprites en series karakters. Stel dat u werkt met 8 sprites, dan is de rest van dat deel van het VRAM beschikbaar om een aantal variabelen op te slaan. Daarbij kunt u bovenstaande getallen als richtlijn nemen.

Voor een sprite-ontwerp heeft u 64 punten ter beschikking, oftewel een rechthoekje van 8 bij 8. 8*8 en toch een rechthoek lijkt paradoxaal, maar een punt is nu eenmaal iets hoger dan hij breed is. Meestal pakt u er een blaadje ruitjespapier bij en geeft de hokjes een kleurtje die u op het scherm wilt zien. Dat zijn de gezette punten, in de andere vakjes verschijnt de achtergrondkleur. Om dit contrastrijke plaatje aan de computer te kunnen voeren dient het te worden omgezet in binaire getallen. Dat is een fluitje van een cent: ingekleurde vakjes zijn gezette punten/bits en dus gelijk aan 1, terwijl de lege hokjes als 0 moeten worden gecodeerd omdat ze overeenkomen met de achtergrondkleur die toch al op het scherm te zien is. Om een sprite daadwerkelijk te laten verschijnen gebruikt u de MSX-BASIC-instructie SPRITE$(N) (N is het nummer van de sprite) in combinatie met de instructie PUT SPRITE. SPRITE$ kan niet overweg met de eerste sprite-codering, de 64 nullen en enen, maar verlangt tekens uit de karakterset. Dat komt goed uit: de 8*8-sprite bestaat uit 8 8-bits binaire getallen onder elkaar en een 8-bits getal kan gehele getallen van 0 tot en met 255 (2 tot de macht 8) weergeven. 8 bits per binair getal (de breedte van de sprite)

komt dus precies overeen met de 255 decimale ASCII-waarden van de karakterset. Anders gezegd: elke combinatie van nullen en enen in een 8-bits getal kan als teken uit de karakterset worden voorgesteld omdat die 255 tekens telt. Dit wat de breedte aangaat. De sprite-hoogte 8 heeft te maken met het feit dat een regel uit 8 lijnen bestaat.

Als we nou eens de binaire representatie van bijvoorbeeld de decimale ASCII-waarde 65 (de A) bekijken

```
PRINT BIN$(65)
```

```
resultaat: 1000001
```

valt op dat er 7 in plaats van 8 bits op het scherm verschijnen. Dat komt omdat voorafgaande nullen worden genegeerd:

```
PRINT BIN$(3)
```

```
resultaat: 11
```

Getallen kleiner dan 8 bits dienen aan de linkerkant dus met nullen te worden aangevuld. Dat gebeurt met de volgende instructie.

```
PRINT STRING$(8-LEN(BIN$(65)),"0")+BIN$(65)
```

Als u dit invoert, kunt u zien hoe de eerste regel van de 8*8-sprite-matrix er uitziet als de toewijzing aan SPRITE$ begint met een A. We nemen nu programma SPRITE 1 onder de loep.

90-160 Hier staat de informatie over de 64 afzonderlijke punten van de sprite in nullen en enen.

30-70 Met deze FOR/NEXT-lus leest de computer achter elkaar de enen en nullen van de 8 spriteregels (regel 40) voegt daar het kenmerk &B aan toe (regel 50). Daarna zet hij ze om in letters en voegt die toe aan de instructie SPRITE$ (regel 60).

Het resultaat van dit programma is dat de tekens uit de karakterset die als code fungeren voor het bitpatroon in de DATA-regels op het scherm verschijnen.

```
SPRITE$(1)="AUAUAUAU"
```

Om de sprite zelf midden op het scherm te plaatsen is een kleine uitbreiding (SPRITE 1a) voldoende. Met PUT SPRITE en het gewenste nummer verschijnt de sprite op het scherm. In het VRAM is ruimte

voor 256 sprites. De grootte van de sprite bepaalt u met de tweede parameter van SCREEN. In SCREEN,0 is de sprite-grootte 8*8 en in SCREEN,1 16*16. In het laatste geval bestaat de sprite nog steeds uit 8*8 punten, maar wordt elke punt opgeblazen tot 2*2 punten. Om het verschil tussen die twee groottes te laten zien voegen we nog enkele regels toe aan het eerste programma (SPRITE 1b). Sprites die echt uit 16*16 onafhankelijke punten bestaan zijn ook mogelijk, maar dan moet u wel een andere schermmodus kiezen. In SCREEN,2 is de sprite-grootte 16*16 en in SCREEN,3 32*32. Ook in dit laatste geval wordt elke punt opgeblazen tot 2*2 punten. Sprites van 16*16 bestaan uit 4 bitpatronen van 8*8, dus uit 4*8 tekens. Die vier rasters worden gelezen in de volgorde linksboven, linksonder, rechtsboven, rechtsonder. Het inlezen wordt er iets ingewikkelder door (zie programma SPRITE 2). Om de 16*16-sprite te kunnen bewonderen moet u nog SCREEN,2 en PUT SPRITE aan het programma toevoegen. Met SCREEN,3 aan het begin van het programma worden 16*16-sprites nog eens tweemaal zo groot. Het is onmogelijk grote sprites samen te persen in een 8*8-matrix met de instructie SCREEN,0 of SCREEN,1. Van de grote sprite blijft in beide gevallen alleen het gedeelte in de linkerbovenhoek over, in SCREEN,0 weergegeven op dezelfde schaal als in SCREEN,2 en in SCREEN,1 tweemaal zo groot.

Hoe wordt nu een 16*16-sprite ingelezen? Daarvoor bekijken we programma SPRITE 2 nog eens. In regel 30 wordt een stringvariabele gedimensioneerd. Daarin wordt de informatie over de vorm van de sprite opgeslagen. Vervolgens worden eerst alle linkerhelften van de sprite-regels bewerkt. Ze worden geïnterpreteerd en bijeengezet in SPRITE$(1). Daarna gebeurt hetzelfde met de 8 rechter sprite-tekencodes. Zo ontstaat uiteindelijk de 16*16-sprite, die eigenlijk bestaat uit vier sprites van 8*8. In het VRAM is ruimte voor 64 sprites van het formaat 16*16, of 256 van het formaat 8*8. Van die 64 kunnen er maar 32 tegelijk op het scherm. Denk er dus goed over na welke keus u maakt.

Opmerkingen

- In het VRAM kunt u lezen waar de sprites zijn opgeslagen. Voor SCREEN 1, 2 en 3 gebruikt u daarvoor respectievelijk BASE(8), BASE(13) en BASE(18). Bij de getallen die daaruit resulteren telt u net zo vaak 8 op als u sprites wilt oproepen. Om in SCREEN 1 de 8 binaire regels van de eerste sprite te lezen gebruikt u programma SPRITE 3. 16*16-sprites rechtstreeks in het VRAM lezen is wat moeilijker. Bekijk programma SPRITE 4. Als u tussentijds de sprite-grootte wilt veranderen, moet u wel goed oppassen. Elke verandering van SCREEN (eerste parameter) of sprite-grootte (tweede parameter

van SCREEN) heeft als resultaat dat alle sprites in het VRAM worden gewist. In principe kan het wel, maar daarvoor moet u veel dieper in uw computer duiken. (zie het programma in paragraaf 9.1).

- U kunt op ieder moment nagaan hoeveel geheugenruimte er nog over is voor sprites.

```
PRINT SPRITE$()
```

geeft het aantal tekens aan dat nog vrij is. Om het aantal vrije sprites te weten te komen moet u het getal delen door 8 als het gaat om 8*8-sprites en door 16 als het gaat om 16*16-sprites.

SPRITE 1

```
10 COLOR 1,15
20 SCREEN 1
30 FOR N=1 TO 8
40 READ A$
50 A$="&B"+A$
60 B$=B$+CHR$(VAL(A$))
70 NEXT N
80 SPRITE$(1)=B$:PRINT SPRITE$(1)
90  DATA 01000001
100 DATA 01010101
110 DATA 01000001
120 DATA 01010101
130 DATA 01000001
140 DATA 01010101
150 DATA 01000001
160 DATA 01010101
```

SPRITE 1a

```
85 PUT SPRITE 1,(128,96)
```

SPRITE 1b

```
15 IF Z=1 THEN Z=0 ELSE Z=1
20 SCREEN 1,Z
25 RESTORE
85 PUT SPRITE 1,(128,96)
86 LOCATE 0,0
87 PRINT "spritegrootte";Z
88 FOR O=1 TO 1000
89 NEXT O:GOTO 15
```

SPRITE 2

```
10 COLOR 1,15
20 SCREEN 1
30 DIM A$(16)
40 FOR N=1 TO 16
```

```
50 READ A$(N)
60 NEXT N
70 FOR N=1 TO 16
80 A$=A$+CHR$(VAL("&B"+LEFT$(A$(N),8)))
90 NEXT N
100 FOR N=1 TO 16
110 A$=A$+CHR$(VAL("&B"+RIGHT$(A$(N),8)))
120 NEXT N
130 SPRITE$(1)=A$
140 DATA 0101010101010101
150 DATA 1010101010101010
160 DATA 0101010101010101
170 DATA 1010101010101010
180 DATA 0101010101010101
190 DATA 1010101010101010
200 DATA 0101010101010101
210 DATA 1010101010101010
220 DATA 0101010101010101
230 DATA 1010101010101010
240 DATA 0101010101010101
250 DATA 1010101010101010
260 DATA 0101010101010101
270 DATA 1010101010101010
280 DATA 0101010101010101
290 DATA 1010101010101010
```

SPRITE 3

```
10 FOR N=14336+32 TO 14336+39
20 PRINT STRING$(8-LEN(BIN$(VPEEK(N))),"0")
+BIN$(VPEEK(N))
30 NEXT N
```

SPRITE 4

```
10 FOR N=14336+32 TO 14336+47
20 PRINT STRING$(8-LEN(BIN$(VPEEK(N))),"0")
+BIN$(VPEEK(N));
30 PRINT STRING$(8-LEN(BIN$(VPEEK(N+16))),"0")
+BIN$(VPEEK(N+16))
40 NEXT N
```

5.9 PUT SPRITE

Het programma PSPRITE 1 staat aan het einde van deze paragraaf.

In het VRAM kunnnen maximaal 256 sprites worden opgeslagen. Daarvan kunnen er 32 tegelijk op het scherm staan. Met de instructie SPRITE$ definiëren we sprites. Met PUT SPRITE kunt u een sprite op het scherm zetten. Met een interrupt gaat de computer na of twee sprites elkaar raken (ON SPRITE GOSUB). Sprites kunnen in alle modi worden weergegeven, behalve in SCREEN 0. In de paragrafen 1.1 en 5.8 hebben we de vier sprite-types en de daarmee samenhangende parameters van SCREEN besproken. De PUT SPRITE-instructie heeft het formaat

```
PUT SPRITE A,(B,C),D,E
```

A

De eerste parameter dient om een hiërarchie aan te brengen in de 32 sprites die op het scherm kunnen komen. Elke sprite krijgt een rangnummer (0-31), een bepaalde prioriteit. U moet het zo zien dat een sprite een hele beeldschermpagina in beslag neemt die op de sprite na doorzichtig is. Sprite nummer 31 is de onderste van de stapel transparante vellen. Op 1 pagina past maar 1 sprite, dus de computer zit pas aan zijn plafond als we sprite 0-31 tegelijk in beeld hebben. Een voorbeeld vindt u in het programma in paragraaf 9.4.

B,C

Met de tweede parameter geeft u de X- en de Y-coördinaten van de sprite aan. Ze mogen waarden hebben tussen -32768 en +32768; een sprite kan dus heel ver buiten beeld zijn. Waarden buiten dit bereik veroorzaken de foutmelding Overflow error. Bij waarden die buiten het bereik van het beeldscherm vallen, verschijnt de sprite om de zoveel tijd (met een getalsinterval) op het scherm. Ook sprites kunnen absoluut en relatief worden geadresseerd. Het middelpunt van het scherm wordt absoluut aangegeven met

```
PUT SPRITE 1,(128,96)
```

Bij relatieve adressering moet u, zoals gewoonlijk, uitgaan van de actuele plaats van de cursor en van daaruit aangeven hoever en in welke richting die zich moet verplaatsen. Met STEP geven we aan dat het om relatieve adressering gaat.

```
PUT SPRITE A,STEP(10,-10),D,E
```

Bedenk wel dat parameter 2 de X- en Y-coördinaat aangeeft van de linkerbovenhoek van het sprite-raster en niet van een punt van de sprite zelf.

D

Met deze parameter kiest u de kleur van de sprite. Iedere sprite mag maar 1 kleur hebben, die u aangeeft met een getal van 0-15. Sprites met meerdere kleuren kunt u alleen maken door sprites met dezelfde vorm, maar met verschillende kleuren elkaar te laten overlappen. Bijvoorbeeld een rood kereltje (sprite 0) met een groen jasje (sprite 1) en een blauwe broek (sprite 2). Zo'n figuur loopt rond in de programma's Freddie en Mr. Miner (paragraaf 9.15). Bij meerkleurige sprites moet u een paar dingen goed in de gaten houden.

- Elkaar overlappende sprites zijn in feite permanent met elkaar in botsing. De meldingen daarvan aan de VDP maken het onmogelijk te werken met de instructie ON SPRITE.
- In BASIC verloopt de besturing van meerkleurensprites soms zo traag dat op het scherm te zien is dat de verschillende kleuren niet tegelijk bewegen. Daar is helaas geen makkelijke oplossing voor.
- Een ander probleem is dat er maximaal 4 sprites op een regel kunnen staan. Als nummer 5 verschijnt, verdwijnt de sprite met het hoogste rangnummer. Het programma in paragraaf 9.5 laat dat zien. Met twee driekleurensprites zit u dus al fout. Toch is het mogelijk met een aantal meerkleurensprites te werken. De spelprogramma's Freddie en Mr. Miner zijn er het bewijs van. Misschien ontdekt u tijdens het intypen ervan hoe dat voor elkaar te krijgen is.

E

Met de laatste parameter geeft u aan welke sprite u wilt oproepen. Let op het verschil met parameter A. Een sprite met een hoog volgnummer (hoge E) kan best een hoge prioriteit (lage A) hebben. De maximale waarde van E hangt af van het type sprite dat u gebruikt. In SCREEN,0 en SCREEN,1 is het maximum aantal sprites 256 en in SCREEN,2 en SCREEN,3 is dat 64. In beide gevallen staan er maximaal 32 sprites tegelijk op het scherm. De sprites op het scherm hoeven er niet verschillend uit te zien; er kunnen ook 32 identieke sprites op het scherm staan (zie programma PSPRITE 1). Ook hier moet u oppassen. Zodra er meer dan 4 sprites op 1 beeldlijn komen (dat is wat anders dan een regel!), verdwijnt het overlappende stuk van de sprite met de laagste prioriteit (de hoogste A). Het programma in paragraaf 9.5 laat dat zien. In de

Y-richting is de enige beperking het maximum aantal sprites dat onder elkaar op het scherm kan staan (32). Als er 32 sprites onder elkaar staan, overlappen deze elkaar wel gedeeltelijk.

Opmerkingen

- Of er meer dan 4 sprites horizontaal naast elkaar staan, kunt u in register VDP(8) lezen (zie paragraaf 1.3). De bits 0-4 van VDP(8) bevatten het nummer van de sprite met de laagste prioriteit. Die gaat dan geheel of gedeeltelijk schuil achter de sprite met een hogere prioriteit.
- In een PUT SPRITE-instructie kunt u de kleur weglaten.

```
PUT SPRITE 1,(128,96),,1
```

Sprite nummer 1 gaat naar positie 128,96 in de actuele voorgrondkleur.

- Om de kleur van een sprite te veranderen hoeft u hem niet op te roepen van het scherm als u de coördinaten van parameter 2 precies kent.

```
DRAW "BM =B;,=C;"
PUT SPRITE 1,STEP(0,0),kleur
```

- Met de instructie BASE kunt u de beginadressen oproepen van twee geheugenbereiken in het VRAM die onmisbaar zijn als we met sprites werken. De sprite attribute table bevat het nummer van de sprite, de actuele coördinaten en de kleur. Deze gegevens hebt u in BASIC nodig voor de instructie PUT SPRITE. De sprite pattern table bevat de gegevens over het uiterlijk van de sprites; die zijn daarin opgeslagen met de BASIC-instructie SPRITE$.
- U kunt ook in plaats van de instructies PUT SPRITE en SPRITE$ rechtstreeks met VPEEK/VPOKE de gewenste geheugenadressen in de hierboven genoemde tabellen aanspreken. PUT SPRITE komt overeen met PRINT VPEEK en SPRITE$ komt overeen met VPOKE. Met PRINT VPEEK komt u bijvoorbeeld de positie en de kleur van sprite 3 aan de weet. Meer informatie over de opbouw van het VRAM staat in de paragrafen over BASE, SCREEN en VDP.
- U kunt een sprite net buiten beeld parkeren met de speciale Y-waarde 209.

```
PUT SPRITE 1,(X,209)
```

De sprite wordt weer zichtbaar zodra hij binnen de beeldgrenzen valt. Een vergelijkbaar effect ontstaat bij Y=208. Maar in dit geval is de verdwijntruc op alle sprites

van toepassing met een lagere prioriteit (grotere A) dan sprite 1. De verdwenen sprites komen terug zodra sprite 1 een zichtbare positie krijgt toegewezen.

PSPRITE 1

```
10 COLOR 1,15
20 SCREEN 1,0
30 VDP(6)=VDP(4)
40 PRINT "eerst 5 sprites met"
50 PRINT "verschillend patroon:"
60 FOR N=0 TO 191
70 PUT SPRITE 1,(128,N+8)
80 PUT SPRITE 2,(128,N+16)
90 PUT SPRITE 3,(128,N+24)
100 PUT SPRITE 4,(128,N+32)
110 PUT SPRITE 5,(128,N+40)
120 NEXT N
130 CLS
140 PRINT "nu 5 sprites met"
150 PRINT "gelijk patroon:"
160 FOR N=0 TO 191
170 PUT SPRITE 1,(128,N+8),,1
180 PUT SPRITE 2,(128,N+16),,1
190 PUT SPRITE 3,(128,N+24),,1
200 PUT SPRITE 4,(128,N+32),,1
210 PUT SPRITE 5,(128,N+40),,1
220 NEXT N
230 RUN
```

6 Geluid

6.1 PLAY

Evenals de groep grafische instructies die samen de Graphics Macro Language (GML) vormen is er een soortgelijke set instructies op het gebied van geluid. Ze heten samen de Music Macro Language (MML). Wat de instructie DRAW voor GML betekent, is PLAY voor MML: met deze instructie worden tonen geproduceerd. De parameters van PLAY vormen samen een string en moeten tussen aanhalingstekens staan. De string mag maximaal 255 tekens lang zijn en bovendien zijn de stringinstructies van toepassing. De verschillende instructie-onderdelen van PLAY hoeven bijna nooit door komma's of spaties van elkaar gescheiden te zijn. Als het echt nodig is, zullen we dat nadrukkelijk aangeven. Vanwege de leesbaarheid staan in dit hoofdstuk steeds wel spaties vermeld. Houd wel in de gaten dat dat geheugenruimte kost. De MSX maakt gebruik van het notensysteem dat we van de middelbare school nog wel kennen (C-D-E-F-G-A-B). Als u

```
PLAY "CDE"
```

invoert, hoort u de tonen C, D en E. Er mogen maximaal 255 noten achter elkaar staan in een string. Om tonen tegelijk te laten klinken nemen we twee strings, gescheiden door een komma.

```
PLAY "CDE","DEF"
```

Zo ontstaan de tweeklanken CD, DE en EF; er mogen weer maximaal 255 noten achter elkaar staan. Een echt akkoord krijgen we pas bij een drieklank en het is voor de MSX geen kunst om een derde muziekkanaal te openen. Ook dit kanaal moet door een komma worden gescheiden van de rest. Uiteraard geldt ook hier de maximum lengte van 255 noten. In totaal kunnen dus 255 akkoorden achter elkaar worden gespeeld.

```
PLAY "C","E","G"
```

Naast de hele tonen C, D, E, F, G, A en B kunnen we ook halve tonen invoeren. Op de piano zijn dat de zwarte toetsen. Halve tonen worden in MML aangegeven met een + (een kruis) of een - (een mol) achter de betreffende hele toon. De complete ladder met alle halve tonen ziet er dan als volgt uit:

```
PLAY "CC+DD+EE+FF+GG+AA+BB+"
```

Wie iets van muziek afweet, ziet dat dit niet klopt want E+ is F en B+ is C. Kijkt u maar naar de afbeelding van het toetsenbord van een piano in het programma in paragraaf 9.9. Elk octaaf bestaat uit zeven hele en vijf halve tonen. De verhoogde E en B bestaan gewoon niet. De C rechts is een octaaf hoger dan de C links. U kunt de halve tonen natuurlijk ook met een - aangeven. We zetten beide mogelijkheden onder elkaar.

```
PLAY "CC+DD+EFF+GG+AA+B"
PLAY "CD-DE-EFG-GA-AB-B"
```

Om het nog meer op muziekschrift te laten lijken mag u ook een kruis (het matje: #) gebruiken voor de notatie van halve tonen. Het kruis moet, net als de + of de -, altijd achter de toon staan waar het betrekking op heeft.

```
PLAY "CC#DD#EFF#GG#AA#B"
```

Al doende hebben we de overbodige halve tonen gevonden. Er is nog een probleem: als we binnen een toonladder een octaaf omhoog willen, moeten we eigenlijk de verhoogde B gebruiken: die zou de volgende C moeten opleveren. Maar in de praktijk blijkt hij gelijk aan de voorgaande (lagere) C. De computer moet dus weten in welke octaaf hij moet werken. Er zijn acht octaven. Standaard werkt de computer in octaaf 4. Andere octaven moet u aangeven met een O gevolgd door het gewenste octaaf.

```
PLAY "CC+DD+EFF+GG+AA+B O5 B+"
```

Mooi voor elkaar, dat wil zeggen de eerste keer. Als u met de cursor omhoog gaat en het deuntje nog eens laat klinken, blijft de muziekchip in octaaf O5 en voert ook netjes de B+ uit in dat octaaf. We horen dus weer de voorgaande C, in plaats van de C die een octaaf hoger ligt. U kunt nu wel O5 veranderen in O6 maar dan blijft u bezig tot het hoogste octaaf is bereikt. Een betere oplossing is op de eerste toon het gewenste etiket te plakken zodat alles herhaalbaar wordt.

```
PLAY "O4 CC+DD+EFF+GG+AA+B O5 B+"
```

Nu is alles voor elkaar en kunt u net zo vaak tussen de octaven heen en weer springen als u wilt. Het maakt de computer niet uit of u voor elke noot het gewenste octaaf vermeldt of alleen als u in een ander octaaf terechtkomt. Voor twee- en drieklanken moet u voor elk van de samenstellende tonen de octaven apart (per kanaal) vermelden; elk kanaal heeft immers zijn eigen string.

```
PLAY "O4 C O5 C O6 C","O4 E O5 E O6 E","O4 G O5 G O6 G"
```

U hoort nu drie keer dezelfde drieklank, maar steeds op een andere hoogte. Laat u de octaafaanduidingen weg, dan blijft u altijd binnen de grenzen van het beginoctaaf (O4). Dan wordt B+ vanzelf een lage C en aan de onderkant wordt C- een hoge(!) B. Leken op muziekgebied mogen in MML getallen gebruiken om de tonen aan te geven. Die getallen moeten dan worden voorafgegaan door een N (number). De laagste C is dan N1 en de B van het achtste octaaf is N96. Een octaaf bestaat uit 12 tonen (zeven hele en vijf halve), dus in totaal zijn er 96 (12*8) tonen. De C in het standaardoctaaf (O4) komt overeen met N36. Het hele octaaf O4 ziet er als volgt uit.

```
PLAY "N36 N37 N38 N39 N40 N41 N42 N43 N44 N45 N46 N47"
```

In vergelijking met het vertrouwde notenschrift zegt dit misschien weinig. Invoer van getallen heeft echter een paar belangrijke voordelen: we hebben niets meer met octaven te maken, want iedere toon heeft een eigen nummer. Voor de computer zijn getallen uiteraard makkelijker te verteren want uiteindelijk wordt alles toch in getalvorm afgehandeld. Zeker als het gaat om stringmanipulaties, heeft de computer veel meer moeite met letters die zich niet houden aan de alfabetische volgorde. In de muziek is naast de toonhoogte ook de toonduur belangrijk. Tot nu toe hebben we daar geen aandacht aan besteed en de MSX ging ervan uit dat u genoegen nam met kwartnoten. De duur van een toon bepaalt u met de parameter L.

```
PLAY "L4"
```

bijvoorbeeld levert een kwartnoot op. L8 is een achtste noot en L1 een hele. U hebt de keus uit de getallen 1, 2, 4, 8, 16, 32 en 64. In het volgende voorbeeld moet u drie kanalen gebruiken; alle drie de deuntjes duren in totaal even lang.

kanaal 1	8 maal 1/8 noot
kanaal 2	4 maal 1/4 noot
kanaal 3	1 maal 1/1 noot

Voor de duidelijkheid staan ze in verschillende octaven.

```
PLAY "O4 L8 CDECDECD","O5 L4 CDECD","O6 L1 C"
```

Net als bij parameter O blijft een gekozen L-waarde geldig tot hij een nieuwe waarde krijgt. Dat is handig als er veel noten van gelijke lengte achter elkaar komen, maar omslachtig als er veel van lengte wordt gewisseld. In dat geval kunt u beter een tweede manier gebruiken om de lengte van een toon aan te geven. U moet dan direct achter de noot de waarde van de lengte zetten.

```
PLAY "C32"
```

is hetzelfde als

```
PLAY "L32 C"
```

Het verschil is dat in het eerste geval de lengte maar voor 1 toon geldt, terwijl in het tweede geval de lengte geldt tot een nieuwe waarde voor L wordt gegeven. Wilt u de lengte van een serie tonen veranderen, dan kunt u de L ook weglaten en in plaats daarvan een puntkomma tussen de waarden voor toonhoogte en toonduur zetten.

```
PLAY "C;8 DE"
```

of

```
PLAY "N36;8 N38 N40"
```

Onder musici is er nog een code gangbaar om de duur van een toon aan te geven. Een punt achter een noot geeft aan dat deze de helft langer moet worden. Een hele toon wordt anderhalve toon en een halve wordt driekwart.

```
PLAY "L4 C."     driekwart noot
PLAY "L1 C."     anderhalve noot
```

Muziek bestaat niet alleen uit tonen, maar ook uit pauzes daartussen. Een pauze noemen we een rust. Dat geeft u aan met R, gevolgd door een getal uit de serie 1-64, net als bij de lengte van de tonen.

```
PLAY "C R4 DE"
```

De toonduur is niet aangegeven; het gaat dus om kwartnoten. De C is van de DE-combinatie gescheiden door een kwart tel rust. Heel korte rustperiodes zijn geschikt om te verhinderen dat een serie gelijke noten klinkt als 1 lange toon.

```
PLAY "CCCC"                   1 lange toon
PLAY "C R64 C R64 C R64 C"    4 aparte aanslagen
```

Het risico van die rusten is dat tonen in de afzonderlijke kanalen uit de pas gaan lopen. Als parameter R geen waarde krijgt, gaat de computer uit van een kwart tel rust als standaardwaarde. De parameters L en R hebben geen absolute waarde: men meet ze niet in seconden. Ze geven alleen de onderlinge lengteverhoudingen aan. De absolute lengte (snelheid) van de tonen hangt af van het tempo waarin ze worden gespeeld. Is het tempo gegeven, dan geldt altijd dat een hele noot gelijk is aan twee halve noten, aan vier

kwartnoten, enzovoort. Het tempo bepaalt u met parameter T. De computer werkt standaard met een tempo van 120. De waarden van T lopen van 32 tot 255. Een T zonder een waarde erachter brengt de computer terug in het basistempo. Beluister de volgende riedeltjes eens.

```
PLAY "T CDE"        tempo 120 : normaal
PLAY "T240 CDE"     tempo 240 : snel
PLAY "T60 CDE"      tempo  60 : langzaam
```

De geluidssterkte van de muziek regelen we met de letter V (volume) gevolgd door een getal tussen 0 (stilte) en 15 (hard). Als u de V weglaat of een V zonder meer invoert gaat de computer uit van V8 (normale geluidsterkte). Vergelijk

```
PLAY "V CDE"       volume  8 : normaal
PLAY "V15 CDE"     volume 15 : hard
PLAY "V1 CDE"      volume  1 : zacht
PLAY "V0 CDE"      volume  0 : uit
```

In de paragraaf over SOUND leggen we uit dat de klank van een toon afhangt van het patroon van de geluidstrilling. U kunt bijvoorbeeld een toon langzaam laten aanzwellen van stilte tot het maximum (crescendo) of hem kort na elkaar harder of zachter laten klinken (vibrato). Hoe een afzonderlijke toon klinkt, hangt af van de omhullende golfvorm, ook wel envelope genoemd. Bij de MSX heeft u de keuze uit acht verschillende patronen. Ze worden aangeduid met een S (shape: vorm, patroon) gevolgd door een getal. Om het effect van zo'n patroon te kunnen waarnemen kunt u het best lange tonen nemen (L1: hele noot); voor de sterkte hoeft u niets te regelen, want zodra u met omhullenden werkt, gaat de computer uit van V16. De verschillende omhullenden hebben de volgende waarden.

```
S1, S4, S8, S10, S11, S12, S13, S14
```

Voer nu bijvoorbeeld

```
PLAY "S1 L1 CDEFG"
```

in en verander daarna de waarde van S. U hoort dan hoe bij iedere omhullende de klank verandert. U kunt het zo gek niet bedenken of de MSX kan het wel nadoen: een helicopter, door de tonen CDE patroon S8 te geven,

```
PLAY "S8 CDE"
```

of trommelslagen met S1

```
PLAY "S1 CDE"
```

Verder kunt u de toon 'voller' maken met de letter M gevolgd door een getal van 1-65535.

```
PLAY "S1 M2000 CDE"
```

De trommelslagen klinken nu wat voller en luider. We zullen dit de 'fijnafstemming' van de omhullende noemen. Het hoe en waarom van deze toevoeging zullen we uitwerken in paragraaf 6.3 bij de instructie SOUND. Nu we alle elementen van de PLAY-programmeertaal hebben gehad, wordt het tijd voor een overzicht.

parameter	betekenis	toegestane waarden
C D E...	namen van de tonen	
+ - #	halve toon hoger of lager	
O	octaaf	1-8
N	nummer van de toon	1-96
L	lengte van de toon	1-64
.	toon 1.5 maal zo lang	
R	rust	1-64
T	tempo	32-255
V	volume	0-15
S	omhullende	1, 4, 8, 10, 11, 12, 13,14
M	fijnafstemming S	1-65535

U heeft vast al gemerkt dat u alweer tonen kunt invoeren als de computer nog bezig is een vlak daarvoor ingetypt deuntje uit te voeren. Er is namelijk een speciaal geheugen dat 25 driestemmige tonen (akkoorden) kan bevatten. Meer dan 25 tonen geeft problemen, want dan kan het lang duren voor het programma met de normale snelheid verder gaat. Meer daarover in paragraaf 6.2 over PLAY(N) en het voorbeeld PLAY(N) 4 aan het eind van die paragraaf. Bij de instructie PLAY kunt u strings aan elkaar koppelen:

```
10 A$="CDE"
20 B$="FGA"
30 PLAY A$,B$
```

of strings invoegen

```
PLAY "CDE X B$;"
```

Een ingevoegde string moet tussen een X en een puntkomma staan. Op dezelfde manier kunt u variabelen invoegen; die moeten tussen een =-teken en een puntkomma staan. In het volgende voorbeeld hoort u de tonen 1-96 achter elkaar.

```
10 FOR N=1 TO 96
20 PLAY "N =N;"
30 NEXT N
```

Om dit niet te lang te laten duren moet u de tonen kort en het tempo hoog houden.

```
20 PLAY "T255 L64 N =N;"
```

Voor het geval u door de bomen het bos niet meer ziet na al deze parameters van PLAY en SOUND: u kunt ze weer op de uitgangswaarden zetten met de instructie BEEP of de toetsencombinatie CTRL-STOP. PLAY, SOUND en BEEP maken alle drie gebruik van dezelfde registerset. Dat heeft meestal tot gevolg dat elk van deze instructies de uitvoering van de andere twee beïnvloedt. Zo zet BEEP alle SOUND-registers terug op de uitgangswaarden.

6.2 PLAY(N)

De programma's PLAY(N) 1-4 staan aan het eind van deze paragraaf.

In de vorige paragraaf hadden we het al even over het muziekgeheugen van de MSX met een omvang van 25 drietonige akkoorden. Nadat u de computer met PLAY hebt laten weten wat u wilt horen, gaat hij verder met het BASIC-programma en voert ondertussen uw compositie uit. Dat is niet zo vanzelfsprekend als het lijkt want er is een interruptbesturing voor nodig. Dat houdt in dat de computer automatisch met regelmatige tussenpozen de uitvoering van het BASIC-programma onderbreekt en controleert of er iets aan de hand is. Daaronder vallen botsingen tussen sprites (ON SPRITE), functietoetsen (als een functietoets wordt gebruikt, treedt ON KEY in werking) en optredende fouten (ON ERROR). U kunt niet alleen horen, maar ook zien hoe het staat met het 25-toons geheugen en de interruptbesturing. Daarvoor dient de instructie PLAY(N). Als via een kanaal een toon klinkt, heeft PLAY(N) de waarde -1; als dat kanaal niet actief is, voert PLAY(N) de waarde 0 uit. Met programma PLAY(N) 1 kunt u zien hoe dat in zijn werk gaat. Bij geluid verschijnt er -1 (waar) op het scherm en zodra de muziek voorbij is komt er een 0 (niet waar) te staan. Met de parameter N kunt u aangeven over welk kanaal u informatie wilt hebben.

PLAY(0)	test op alle drie kanalen
PLAY(1)	test op kanaal 1
PLAY(2)	test op kanaal 2
PLAY(3)	test op kanaal 3

Het nut van deze test blijkt duidelijk uit programma PLAY(N) 2, waarin op kanaal 1 de muziek nog vrolijk doorgaat terwijl de andere kanalen al stilgevallen zijn. Het programma zorgt ervoor dat dat ook op het scherm te zien is. Successievelijk veranderen de vier 'aan'-meldingen in 'uit'. Misschien ten overvloede de toelichting dat de IF...THEN-voorwaarden in de regels 90-120 ervoor zorgen dat als A waar is (A = -1 betekent dat er minstens 1 kanaal actief is) het woord 'aan' op het scherm moet komen te staan. De computer kijkt of volgens de regels van de Boole-algebra is voldaan aan de voorwaarde en voert dan al of niet de opdracht uit. Door de invoer een beetje handig in te delen moet het lukken het muziekgeheugen zo vol te houden dat de uitvoering van de muziek niet stopt bij gebrek aan invoer. Of de computer inderdaad 25 tonen op eigen houtje weergeeft, komt u aan de weet als u programma PLAY(N) 3 draait. Vanaf geheugenplaats 63830 in het RAM ligt het bereik met de gegevens over de PLAY-instructie. Bedenk een manier om uit te vinden wat zich daar allemaal afspeelt tijdens de uitvoering van een melodie. Een stukje van die puzzel vormt het gegeven dat u de kanalen afzonderlijk kunt laten zwijgen.

kanaal 1 uitschakelen

```
POKE 63833,255        POKE 63834,255
```

kanaal 2 uitschakelen

```
POKE 63839,255        POKE 63840,255
```

kanaal 3 uitschakelen

```
POKE 63845,255        POKE 63846,255
```

Omdat deze geheugenplaatsen de waarde 0 bevatten als er geen geluid is, kunt u in plaats van de instructie PLAY(N) ook deze adressen oproepen. Als u zo'n testlus gebruikt, heeft dat ook nog het voordeel dat u automatisch te zien krijgt hoeveel tonen er op ieder kanaal al zijn gespeeld en hoeveel er in totaal op het programma staan (zie programma PLAY(N) 4).

PLAY(N) 1
```
10 COLOR 1,15
20 SCREEN 0
30 PLAY "cdefgab"
40 PRINT PLAY(0)
50 GOTO 40
```

PLAY(N) 2
```
10 COLOR 1,15
20 SCREEN 0
30 WIDTH 30
40 PLAY "cdefgab","defgab","efgab"
50 A=PLAY(0)
60 B=PLAY(1)
70 C=PLAY(2)
80 D=PLAY(3)
90 IF A THEN PRINT "AAN  ";ELSE PRINT "UIT  "
100 IF B THEN PRINT "AAN  ";ELSE PRINT "UIT  "
110 IF C THEN PRINT "AAN  ";ELSE PRINT "UIT  "
120 IF D THEN PRINT "AAN  ";ELSE PRINT "UIT  "
130 GOTO 50
```

PLAY(N) 3
```
10 CLEAR 2000
20 COLOR 1,15
30 SCREEN 0
40 WIDTH 37
50 FOR N=1 TO 25
60 A=INT(RND(1)*96)
70 B=INT(RND(1)*96)
80 C=INT(RND(1)*96)
90 A$=A$+"N"+STR$(A)
100 B$=B$+"N"+STR$(B)
110 C$=C$+"N"+STR$(C)
120 NEXT N
130 PLAY A$,B$,C$
140 PRINT"25 tonen op 3 muziekkanalen spelen vanaf
nu zelfstandig; u kunt ondertussen verder programmeren
(of telt u eens rustig mee)"
```

PLAY(N) 4
```
10 COLOR 1,15
20 SCREEN 0
30 WIDTH 37
40 PLAY "cdedcdecodeodeodeodeodec","cededeodededed
edcdede","cdededededededede"
50 LOCATE0,0
60 PRINT INT(PEEK(63834!)/5)"e toon uit kanaal 1"
70 LOCATE 0,1
80 PRINT (PEEK(63833!)-1)/5"Tonen komen uit kanaal 1"
90 PRINT
100 PRINT INT(PEEK(63840!)/5)"e toon uit kanaal 2"
110 PRINT (PEEK(63839!)-1)/5"tonen komen uit kanaal 2"
```

```
120 PRINT
130 PRINT INT(PEEK(63846!)/5)"e toon uit kanaal 3"
140 PRINT (PEEK(63845!)-1)/5"tonen komen uit kanaal 3"
150 PRINT
160 IF PEEK(63833!)=PEEK(63834!) THEN PRINT
 "kanaal 1 is uit" ELSE PRINT "kanaal 1 is aan"
170 IF PEEK(63839!)=PEEK(63840!) THEN PRINT
 "kanaal 2 is uit" ELSE PRINT "kanaal 2 is aan"
180 IF PEEK(63845!)=PEEK(63846!) THEN PRINT
 "kanaal 3 is uit" ELSE PRINT "kanaal 3 is aan"
190 GOTO 50
```

6.3 SOUND

Er zijn diverse methoden om geluiden te ontlokken aan de MSX. BEEP bijvoorbeeld geeft een waarschuwingssignaal, zoals wanneer er een fout optreedt (meer daarover in paragraaf 6.4). De toetsklik schakelt u in met de derde parameter van SCREEN. In de vorige paragraaf hebben we gezien hoe we met PLAY driestemmig kunnen musiceren. Met de instructie SOUND spreken we rechtstreeks de registers van de PSG (programmable sound generator) aan. Er zijn 14 registers, die we een voor een langslopen.

register 0

Met register 0 geeft u de hoogte van de toon op kanaal 1 aan. Toegestane waarden: 0-255. Voorwaarde is dat met register 7 kanaal 1 is ingeschakeld en bovendien met register 8 het volume is ingesteld. Voorbeeld: SOUND 0,200.

register 1

Net als register 0 heeft register 1 te maken met de toonhoogte in kanaal 1. Alleen is de instelling grover. Een stap in register 1 betekent een octaaf verschil. De octaven zijn genummerd van 0-15. Voorbeeld: SOUND 1,10.

U moet de registers 0 en 1 beschouwen als een registerpaar met respectievelijk een fijne en een grove afstemming van de toonhoogte. Het bovenstaande verhaal geldt ook voor de registerparen (2,3) en (4,5) van achtereenvolgens de kanalen 2 en 3.

register 2

Fijnafstemming van de toonhoogte in kanaal 2 (toegestane waarden: 0-255).

register 3

Grove afstemming van de toonhoogte in kanaal 2 (toegestane waarden: 0-15).

register 4

Fijnafstemming van de toonhoogte in kanaal 3 (toegestane waarden: 0-255).

register 5

Grove afstemming van de toonhoogte in kanaal 3 (toegstane waarden: 0-15).

Ook voor deze en volgende registers gelden de bij register 0 genoemde voorwaarden. Bij register 7 hebt u de keus uit toon, ruis, toon + ruis of stilte. Om tonen te horen moet u dit register uiteraard instellen op 'toon'. De registers 8, 9 en 10 dienen om de geluidssterkte in te stellen van respectievelijk de kanalen 1, 2 en 3.

register 6

Als u met register 7 hebt gekozen voor ruis op een van de kanalen, kunt met register 6 kiezen uit acht verschillende typen ruis (waarden 0-7). Voorbeeld: SOUND 6,5.

register 7

De waarden van dit register moeten binair worden ingevoerd. Voorbeeld: SOUND &B10101010. We hebben deze schrijfwijze gekozen omdat zo de functie van dit register het duidelijkst tot uiting komt. De bits 0-2 (geteld vanaf de rechterkant) geven aan op welke kanalen tonen klinken; in het voorbeeld is dat kanaal 2. De twee nullen betekenen dat de kanalen 1 en 3 stil zijn. De volgende drie bits (3-5: 101) bepalen welke kanalen ruis moeten geven, in ons voorbeeld dus de kanalen 1 en 3. De bits 6 en 7

moeten altijd respectievelijk 0 en 1 zijn en hebben verder geen betekenis. Nog een voorbeeld.

```
SOUND 7,&B10100110

          110    tonen op kanaal 2 en 3
        100      bovendien ruis op kanaal 3
      10         ---
```

register 8

Hiermee bepaalt u het volume van de toon of de ruis op kanaal 1. Toegestane waarden: 0-15. Hoger mag ook maar de toon wordt er niet harder van. Een grotere waarde zorgt voor een vervorming die u verder kunt instellen met de registers 11-13. Een omhullende (S) werkt pas als de geluidssterkte groter is dan 15. Voorbeeld: SOUND 8,30.

register 9

Hiervoor geldt hetzelfde als voor register 8, maar gaat het om kanaal 2.

register 10

Idem, maar dan voor kanaal 3.

Om op kanaal 1 tonen, op kanaal 2 ruis en op kanaal 3 een golfvorm van een eerder gekozen toon te krijgen, is het volgende rijtje instructies nodig.

```
SOUND 1,10
SOUND 5,12
SOUND 6,4
SOUND 7,&B10010101
SOUND 8,10
SOUND 9,10
SOUND 10,10
```

register 11 en 12

Hiermee regelt u de frequentie van omhullenden. Dit registerpaar werkt op dezelfde manier als bijvoorbeeld register 0 en 1. Register 11 dient voor de fijnafstemming van de frequentie en

register 12 voor de grove afstemming. U moet wel eerst een omhullende kiezen (register 13). De toegestane waarden van beide registers zijn 0-255. In totaal zijn er 65.535 frequenties mogelijk. Voorbeeld: SOUND 11,11 en SOUND 12,42.

register 13

De aard van een toon hangt sterk af van het type omhullende. Een claxon bijvoorbeeld geeft een constante toon, terwijl bij een sirene de sterkte van de toon voortdurend toe- en afneemt. De geluids-chip van de MSX heeft nog veel meer mogelijkheden in petto.

0 de toon is meteen maximaal en sterft langzaam weg

4 de toon zwelt aan tot een maximum en breekt dan abrupt af

8 de toon wordt zachter, bereikt dan direct het maximum, wordt weer zachter, enzovoort

10 de toon wordt langzaam zachter, dan weer harder, dan weer zachter, enzovoort

11 de toon wordt zachter en is na een moment stilte plotseling op maximale sterkte (zonder stijging of daling)

12 de toon zwelt aan, valt ineens weg, neemt weer toe, enzovoort

13 de toon komt langzaam op en blijft dan op maximale sterkte

14 de toon komt langzaam op, zakt langzaam weg, komt langzaam weer op, enzovoort (net als bij nummer 10, maar daar is de sterkte aan het begin maximaal)

Voor de overzichtelijkheid vatten we functies van de registers nog eens samen.

register	**omschrijving**	**waarden**
0	fijnafstemming van de freqentie op kanaal 1	0-255
1	grove afstemming van de frequentie op kanaal 1	0-15

2	fijnafstemming van de frequentie op kanaal 2	0-255
3	grove afstemming van de frequentie op kanaal 2	0-15
4	fijnafstemming van de frequentie op kanaal 3	0-255
5	grove afstemming van de frequentie op kanaal 3	0-15
6	soorten ruis	0-7
7	laagste drie bits: wel/geen toon (1 of 0) op kanaal 3/2/1 volgende drie bits: wel/geen ruis (1 of 0) op kanaal 3/2/1 hoogste twee bits: altijd 10 (geen functie)	
8	geluidssterkte op kanaal 1 omhullende	0-15 16
9	geluidssterkte op kanaal 2 omhullende	0-15 16
10	geluidssterkte op kanaal 3 omhullende	0-15 16
11	fijnafstemming van de frequentie van de omhullende	0-255
12	grove afstemming van de frequentie van de omhullende	0-15
13	keuze van de omhullende	0/4/8/ 10-14

Het is niet voldoende om met register 13 een omhullende te kiezen. Al naar gelang het gekozen geluid moet u verschillende registers gebruiken. Een paar voorbeelden.

Een gewone toon op kanaal A

1 toonhoogte instellen met register 0
2 eventueel toonhoogte instellen met register 1
3 in register 7 bit 0 de waarde 1 geven
4 voor het volume register 8 een waarde tussen 0 en 15 geven

Ruis op kanaal B

1 in register 6 het type ruis kiezen
2 in register 7 bit 4 de waarde 1 geven
3 voor het volume register 9 een waarde tussen 0 en 15 geven

Een loeiende sirene op kanaal C

1 toonhoogte instellen met register 4
2 eventueel toonhoogte instellen met register 5
3 in register 7 bit 2 de waarde 1 geven
4 register 10 op 16 zetten (voor omhullenden)
5 de frequentie van de omhullende regelen met register 11 en/of 12
6 een geschikt patroon voor de omhullende kiezen in register 13; waarschijnlijk is dat nummer 10 of 14

Geluid produceren met de instructie SOUND heeft het grote voordeel dat een geluid, eventueel zelfs driestemmig, blijft klinken tot u het zelf afbreekt. Het is dus heel gemakkelijk om bij muziek voortdurend op de achtergrond ruis te laten klinken of tijdens een spel steeds het geronk van een helicopter te horen. Opmerkingen:

- Als u met register 7 ruis hebt ingevoerd op een kanaal en vervolgens probeert met de instructie PLAY uit datzelfde kanaal een toon te laten komen, geeft de MSX aan dat hij de invoer heeft geregistreerd. U hoort echter geen toon, maar ruis.
- Met de volgende formules kunt u de toonfrequenties van de verschillende kanalen uitrekenen.

kanaal A	124797/256*(register 1)+(register 2)
kanaal B	124797/256*(register 3)+(register 4)
kanaal C	124797/256*(register 5)+(register 6)

- Hoewel u van alles in de 14 sound-registers kunt schrijven, kunt u de daar opgeslagen informatie niet oproepen. Met de instructie PLAY(N) komt u in ieder geval iets te weten: door voor N een getal van 0-3 in te vullen, gaat de computer na of een kanaal actief is (-1) of niet (0).

6.4 BEEP

De programma's BEEP 1-3 staan aan het einde van deze paragraaf.

Bij de instructie BEEP lopen menig programmeur de rillingen over de rug omdat dit geluid ook optreedt als de computer een fout constateert en er een boodschap op het scherm verschijnt (bijvoorbeeld Syntax error in 10). Zo'n melding kunnen we snel genoeg uitlokken door zonder regelnummers wat tekens in te voeren gevolgd door RETURN. Er zijn ook nettere manieren om de pieptoon te produceren.

1 PRINT CHR$(7)
2 tegelijk de toetsen CTRL en G indrukken
3 PLAY "L64 M255 06 S1 T120 V8 E"
4 programma BEEP 1

Zoals u ziet krijgt in programma BEEP 1 register 7 een decimale waarde toegewezen. Om te zien welke kanalen daardoor in actie komen moet u die waarde omzetten in binaire vorm. Er zijn 6 bits beschikbaar, dus zijn de decimale waarden 0-63 toegestaan. Al met al hebben we al zes manieren om deze overbekende pieptoon op te roepen (de instructie BEEP zelf en de foutmelding meegerekend). Hoewel ze hetzelfde klinken, hebben ze zeker niet dezelfde betekenis. Luister maar eens naar programma BEEP 2. Om eventuele verschillen met de andere piepers vast te stellen, zet u in regel 20 de instructies die we net hebben genoemd, behalve CTRL + G (programma BEEP 1 kunt u invoegen in de regels 20-25). U heeft waarschijnlijk wel gemerkt dat door de instructies BEEP en PRINT CHR$(7) de registers van PLAY en SOUND worden teruggezet op de uitgangstoestand. Als u de instructie PLAY of het SOUND-programma gebruikt, worden de registers ongemoeid gelaten. Om geen informatie over geluiden kwijt te raken, moet u het waarschuwingssignaal opwekken met mogelijkheid 3 of 4. Op die manier houdt u alle sound-registers ter beschikking. Onverhoeds optreden van de fout-piep kunt u voorkomen door in het begin van het programma met ON ERROR een interruptbesturing in te schakelen. Om misverstanden te vermijden vermelden we dat waarschuwingssignalen niet beperkt hoeven te blijven tot fouten. Programma BEEP 3 geeft daar een voorbeeld van. Zodra u tegen de spelregels zondigt, wordt u met een pieptoon op de vingers getikt.

```
BEEP 1     10 SOUND 0,85
           20 SOUND 1,0
           30 SOUND 7,8
           40 SOUND 8,7
           50 FOR N=1 TO 10
           60 NEXT N
           70 SOUND 7,9

BEEP 2     10 PLAY "O6 CEG"
           20 FOR N=1 TO 1000
           30 NEXT N
           40 BEEP
           50 PLAY "CEG"

BEEP 3     10 COLOR 1,15
           20 SCREEN 0
           30 PRINT "Voer een getal"
           40 INPUT "tussen 0 en 50 in";A
           50 IF A<0 OR A>50 THEN BEEP:GOTO 30
           60 PRINT "Goed gedaan"
           70 GOTO 30
```

7 De joystick-aansluitingen

7.1 STICK

De programma's STICK 1-3 staan aan het einde van deze paragraaf.

De MSX heeft twee joystick-ports. De naam van deze instructie geeft al aan dat hij betrekking heeft op een eventueel aangesloten joystick. Met STICK kunt u de richting uitlezen die de joystick aangeeft. Met een aparte parameter geeft u daarbij aan welk van de drie volgende mogelijkheden wordt gewenst.

PRINT STICK(0)	cursortoetsen uitlezen
PRINT STICK(1)	joystick op poort 1 uitlezen
PRINT STICK(2)	joystick op poort 2 uitlezen

Zo kunnen er dus drie spelers tegelijk met een programma bezig zijn: twee met een joystick en nog een met de cursortoetsen. Is er maar 1 speler, dan laat de parameter van STICK deze de keuze tussen joystick of cursortoetsen. Het toetsenbord is ook geschikt als start-/vuurknop; daarvoor moet u de spatiebalk gebruiken. Met andere woorden: de interruptbesturing na ON STRIG reageert ook op de spatiebalk. De instructie STICK maakt onderscheid tussen 8 richtingen en de ruststand. Daarbij gelden de volgende waarden:

0 geen richting (uitgangstoestand, joystick in middenpositie)

1 rechts omhoog (windrichting NO)
cursortoetsen: pijl omhoog + pijl naar rechts

2 naar rechts (windrichting O)

3 rechts omlaag (windrichting ZO)
cursortoetsen: pijl naar rechts + pijl omlaag

4 omlaag (windrichting Z)

5 links omlaag (windrichting ZW)
cursortoetsen: pijl omlaag + pijl naar links

6 naar links (windrichting W)

7 links omhoog (windrichting NW)
cursortoetsen: pijl naar links + pijl omhoog

8 omhoog (windrichting N)

Als u twee tegenover elkaar liggende cursortoetsen indrukt, negeert de computer dat. Drukt u er drie tegelijk in, dan zijn er altijd twee die elkaar opheffen en blijft de derde vanzelf over. De combinatie links/omlaag/rechts levert zo de richting 'omlaag' op met de daarbij behorende waarde 4. Het eerste voorbeeld, programma STICK 1, is eigenlijk een tekenprogramma waarbij u met de joystick tekent. Bekijk eerst maar eens hoe het werkt. Mocht u de beweging van de tekencursor te langzaam vinden, dan kunt u de stapgrootte veranderen. De kompasrichting NO (B = 2) zou bijvoorbeeld kunnen resulteren uit PSET STEP (+2,-2). Het maakt de MSX niet uit of u buiten het schermbereik (256*192 punten) terechtkomt. Het nadeel is wel dat u op die manier het overzicht kwijtraakt. Het is dan ook aan te raden in uw programma's een maximum aan te houden voor de coördinaten. Dat kan bijvoorbeeld door ze de waarde 0 te geven zodra ze groter dan 256 of 192 zijn (de randen van het scherm). Als u dan een lijn tekent, verdwijnt die niet uit beeld maar komt aan het begin van dezelfde regel of kolom terug. Programma STICK 2 is bedoeld om na te gaan welke richtingen de joystick aankan. Daarmee kunt u de kwaliteit van de joystick testen. Bij het grote aanbod van tegenwoordig, is dat geen overbodige luxe. Met STICK 2 kunt u natuurlijk ook de cursortoetsen uitproberen, maar dat kunt u ook rechtstreeks doen, bijvoorbeeld in een stukje tekst. Net als de andere toetsen heeft elke cursortoets een code. Dus is er naast PRINT STICK(0) nog een tweede manier om achter de werking van de cursortoetsen te komen: PRINT ASC(A$). Programma STICK 3 laat dat zien. De cursortoetsen hebben de volgende codes.

rechts	CHR$(28)
links	CHR$(29)
boven	CHR$(30)
onder	CHR$(31)

Programma STICK 3 kan niet goed uit de voeten met de 'tussenrichtingen': in die gevallen ziet u op het scherm een reeks getallen die afwisselend de beide ingedrukte toetsen weergeven.

STICK 1

```
10 SCREEN 0
20 COLOR 1,15
30 WIDTH 40
40 PRINT "Toetsenbord        = 0
          Joystick-port 1  = 1
          Joystick-port 2  = 2"
50 PRINT
60 INPUT "Uw invoer (0 t/m 2) ";A
70 IF A<0 OR A>2 THEN RUN
80 SCREEN 2
90 PSET(128,96)
100 B=STICK(A)
110 IF B=1 THEN PSET STEP (0,-1)
120 IF B=2 THEN PSET STEP (+1,-1)
130 IF B=3 THEN PSET STEP (+1,0)
140 IF B=4 THEN PSET STEP (+1,+1)
150 IF B=5 THEN PSET STEP (0,+1)
160 IF B=6 THEN PSET STEP (-1,+1)
170 IF B=7 THEN PSET STEP (-1,0)
180 IF B=8 THEN PSET STEP (-1,-1)
190 GOTO 100
```

STICK 2

```
10 SCREEN 0
20 COLOR 1,15
30 WIDTH 40
40 PRINT "Toetsenbord        = 0
          Joystick-port 1  = 1
          Joystick-port 2  = 2"
50 PRINT
60 INPUT "Uw invoer (0 t/m 2) ";A
70 IF A<0 OR A>2 THEN RUN
80 ON STRIG GOSUB 240,240,240,240,240
90 FOR N=0 TO 4
100 STRIG(N) ON
110 NEXT N
120 SCREEN 2
130 B=STICK(A)
140 IF B=0 THEN GOTO 130: ELSE PSET (128,96)
150 IF B=1 THEN DRAW "u16 n f8 g8"
160 IF B=2 THEN DRAW "e16 n d8 l8"
170 IF B=3 THEN DRAW "r16 n h8 g8"
180 IF B=4 THEN DRAW "f16 n u8 l8"
190 IF B=5 THEN DRAW "d16 n h8 e8"
200 IF B=6 THEN DRAW "g16 n u8 r8"
210 IF B=7 THEN DRAW "l16 n e8 f8"
220 IF B=8 THEN DRAW "h16 n d8 r8"
230 GOTO 130
240 CLS
250 RETURN
```

```
STICK 3   10 SCREEN 0
20 COLOR 1,15
30 WIDTH 40
40 PRINT "De cursorbesturingstoetsen testen"
50 A$=INKEY$
60 IF A$="" THEN GOTO 50
70 LOCATE 10,10
80 IF ASC(A$)=28 THEN PRINT "Rechts"
90 IF ASC(A$)=29 THEN PRINT "Links "
100 IF ASC(A$)=30 THEN PRINT "Boven "
110 IF ASC(A$)=31 THEN PRINT "Onder "
120 GOTO 50
```

7.2 STRIG

De programma's STRIG 1 en 2 staan aan het einde van deze paragraaf.

De instructie STRIG doet niet meer dan constateren of de drukknop(pen) van de joystick of de spatiebalk wel/niet zijn ingedrukt. U hebt nu niet meer te maken met irritant lange programmalussen als u erachter wilt komen of een bepaalde knop/toets is ingedrukt. Dat kan nu via interruptbesturing na ON STRIG GOSUB. U moet wel specificeren over welke knop of toets u informatie wilt. De parameter van STRIG(N) mag de waarden 0-4 hebben. De betekenis hiervan is:

0	spatiebalk
1	joystick 1, knop 1
2	joystick 2, knop 1
3	joystick 1, knop 2
4	joystick 2, knop 2

U moet wel in de gaten houden dat u alleen een tweede knop kunt uitlezen als de joystick volledig voldoet aan de MSX-norm, waarbij geldt dat de twee knoppen helemaal onafhankelijk van elkaar moeten werken. Als u een spel wilt maken met de bedoeling daar uw brood mee te verdienen, moet u er wel voor zorgen dat in het beginmenu de mogelijkheid aanwezig is om ook met een normale joystick (met 1 drukknop) te spelen waarbij de speler toch alle mogelijkheden van het programma moet kunnen uitbuiten. Als uw spel geschikt is voor drie spelers moet u eerlijkheidshalve geen gebruik maken van de mogelijkheden van een tweede knop omdat de derde speler slechts de beschikking heeft over 1 vuurknop, de spatiebalk.

STRIG 1

```
10 SCREEN 0
20 ON STRIG GOSUB 70,90,110,130,150
30 FOR N=0 TO 4:STRIG(N) ON
40 STRIG(N) ON
50 NEXT N
60 GOTO 60
70 PRINT "Spatiebalk ingedrukt"
80 RETURN
90 PRINT "Joystick 1, drukknop ingedrukt"
100 RETURN
110 PRINT "Joystick 2, drukknop 1 ingedrukt"
120 RETURN
130 PRINT "Joystick 1, drukknop 2 ingedrukt"
140 RETURN
150 PRINT "Joystick 2, drukknop 2 ingedrukt"
160 RETURN
```

STRIG 2

```
10 SCREEN 0
20 PRINT "Spatiebalk indrukken"
30 PRINT
40 A$=INKEY$
50 IF A$="" THEN GOTO 40
60 IF ASC(A$)=32 THEN PRINT "Spatiebalk ingedrukt"
70 GOTO 40
```

7.3 PAD

De programma's PAD 1-3 staan aan het einde van deze paragraaf.

De MSX heeft niet alleen twee joystick-ports, maar ook een hele serie BASIC-instructies waardoor u deze bussen optimaal kunt benutten. Een van die instructies is PAD. Deze dient om invoer via een 'graphics tablet' of 'pad' (een grafisch tableau) uit te lezen. Dat is een plaat met verschillende lagen, waarvan de belangrijkste bestaat uit drukgevoelig materiaal. Als u met een pen of een ander puntig voorwerp op de plaat drukt, veroorzaakt dat een signaal. Dat gaat naar de joystick-port en geeft aan de computer de plaats door waar het contact is gesignaleerd. Daarbij gebruikt de computer hetzelfde coördinatensysteem als voor het beeldscherm. Net als bij een joystick zit er op zo'n tableau ook een drukknop om informatie aan de computer door te geven. Ook dit gegeven kunt u uitlezen met de instructie PAD. Bij de MSX wordt niet standaard een grafisch tableau geleverd; in de meeste gevallen zult u dat apart moeten bestellen. De bedoeling van een grafisch tableau is natuurlijk dat wat u erop schrijft of tekent ook zichtbaar wordt op het beeldscherm. Een fraai voorbeeld van de mogelijkheden is het volgende. U voert eerst een bepaalde schaalgrootte in: met hoeveel kilometer komt een bepaalde lijnlengte overeen. Dan legt u een kaart op het tableau en trekt met een pen een willekeurige weg over (bijvoorbeeld van Groningen naar Maastricht). Die weg komt ook op het scherm te staan en de MSX berekent de afstand Groningen-Maastricht. De knop op het tableau kunt u voor van alles gebruiken: overstappen van het ene naar het andere plaatje of van een plaatje naar een menu (bijvoorbeeld een HELP-menu). Nog een mogelijkheid is om er een andere tekenkleur mee te kiezen. De toepassingsmogelijkheden lijken onbegrensd, maar in de praktijk wordt PAD meestal gebruikt om mee te tekenen en de drukknop om te kiezen tussen een plaatje en een menu. Jammer genoeg heeft de MSX geen ingebouwd programma om een tableau te kunnen uitlezen. Daar moeten we dus zelf voor zorgen. Met PRINT PAD(N) kunt u alle verschillende waarden op het scherm zetten. Voor N vult u een getal van 0-7 in. De betekenis daarvan is:

joystick-port 1: PAD(0) - PAD(3)
joystick-port 2: PAD(4) - PAD(7)

0/4	tableau
1/5	X-coördinaat
2/6	Y-coördinaat
3/7	drukknop

Het is niet voldoende alleen de variabele PAD(0) of PAD(4) te lezen want de enige informatie die u dan krijgt, is of er wel (-1)

of niet (0) wat op het tableau gebeurt. U kunt PAD(0) en PAD(4) opvatten als schakelaars waarmee u de tableau-uitlezing aanzet. Als er een contact op het tableau is geregistreerd, krijgt PAD(0) of PAD(4) de waarde -1. Op dat moment krijgen de variabelen voor de coördinaten (PAD 1/2 en/of 5/6) hun actuele waarde (zie programma PAD 1). Tot het moment waarop een nieuw contact wordt geregistreerd, blijven de oude waarden op het scherm zichtbaar. Is er nog niets geregistreerd, dan wacht het programma daarop (regel 40). Stel, u drukt op het midden van het tableau (128,96). PAD(0) heeft de waarde -1 zolang de stift op het tableau drukt. PAD(1) krijgt de actuele X-waarde, die u met PRINT PAD(1) kunt opvragen. Hetzelfde geldt voor PAD(2) met betrekking tot de Y-waarde. Is het tableau aangesloten op joystick-port 2, dan gaat het vanzelfsprekend om PAD(4), (5) en (6). PAD(3)/PAD(7) geven aan of er een signaal van de drukknop is binnengekomen; zo ja, dan heeft deze variabele de waarde -1 (= waar) en anders 0 (= niet waar). In het eerste geval is het resultaat een verandering, bijvoorbeeld overschakeling naar een ander beeld. Laten we programma PAD 2 eens nader bekijken. Zolang er geen contact is geregistreerd en PAD(0) de waarde 0 heeft, blijft het programma bij regel 30 hangen. Zodra het tableau wordt aangeraakt, gaat het programma naar regel 40. Die spreekt voor zich. In regel 50 wacht het programma tot de drukknop is ingedrukt en in regel 60 keert het terug naar het begin. Programma PAD 3 demonstreert hoe u met het tableau kunt tekenen. Hierin wordt niet doorlopend gecontroleerd of er op het tableau wordt gedrukt of niet, maar wordt steeds opnieuw met PSET de actuele grafische cursor gezet (regel 60). Het gaat hier om de drukknop, waarmee u de voorgrondkleur (zwart) en de achtergrondkleur (wit) kunt verwisselen (regel 30). Met een beetje geduld kunt u zo een tekening op het tableau maken en direct overbrengen naar het scherm. Bovendien kunt u stukjes voorgrond uitwissen door de knop in te drukken en de gewenste stukjes nog een keer met de achtergrondkleur te tekenen. Tenslotte vermelden we nog de adressen waar de coördinaten worden opgeslagen: de X-waarde op adres &HFC9C en de Y-waarde op adres &HFC9D.

PAD 1

```
10 COLOR 1,15
20 SCREEN 0
30 A=PAD(0)
40 IF A THEN GOTO 50 ELSE GOTO 30
50 PRINT PAD(1)
60 PRINT PAD(2)
70 GOTO 30
```

PAD 2

```
10 COLOR 1,15
20 SCREEN 0
30 IF PAD(0)=0 THEN GOTO 30
40 PRINT PAD(1),PAD(2)
50 IF PAD(3)=0 THEN GOTO 50
60 GOTO 30
```

PAD 3

```
10 COLOR 1,15
20 SCREEN 0
30 IF PAD(3)=-1 THEN IF A=2 THEN A=1 ELSE A=2
40 IF A=2 THEN COLOR 15
50 IF A=1 THEN COLOR 1
60 PSET(PAD(1),PAD(2))
70 GOTO 30
```

7.4 PDL

De programma's PDL 1 en 2 staan aan het einde van deze paragraaf.

Het volgende apparaat dat we kunnen aansluiten op de joystick-port is de zogenaamde 'paddle'. Dat is een soort joystick, met dien verstande dat u een paddle alleen als draaiknop kunt gebruiken waarbij er maar 1 draairichting is (rechts-links of omhoog-omlaag). Een voorbeeld: u bent acrobaat in een circus. Uw taak is een wip die in de piste staat heen en weer te bewegen. Dat doet u met de paddle. Uit de nok van de tent springt uw partner naar beneden. Het is de bedoeling dat deze zo op de wip terechtkomt dat hij weer in de nok belandt. Als u zo'n programma voor de MSX zou willen schrijven, is programma PDL 1 geschikt als subroutine om de paddle te besturen. Al naar gelang de draairichting van de paddle produceert dit programma waarden tussen 0 en 255, waarmee we sprites rechtstreeks horizontaal kunnen besturen. Eigenlijk is PDL een gedimensioneerde variabele in plaats van een instructie. De waarde van die variabele wordt echter voortdurend aangepast aan de stand van de paddle. PDL bevat een parameter, waarmee we de twee joystickports van elkaar kunnen onderscheiden. Die parameter moet altijd worden vermeld. De oneven getallen 1-11 hebben betrekking op de stand van de

eerste paddle en de even getallen 2-12 op de stand van de tweede paddle. Waarden buiten dit bereik veroorzaken de foutmelding Illegal function call. De uitvoering van het programma wordt dan onderbroken.

Opmerkingen

- Een joystick-port uitlezen met PDL duurt ongeveer 0.01 seconde. Dat is nogal lang voor een MSX-BASIC-instructie. Hij is daardoor geschikt om als wachtlus te gebruiken.
- Als u de waarde van de paddle opvraagt zonder dat er een is aangesloten op de computer, krijgt u als antwoord de waarde 255.
- Als u bij een joystick de laatst ingevoerde waarde opvraagt, is dat niet de laatse positie waar u de cursor heen hebt gestuurd, omdat de stuurknuppel direct als u hem loslaat naar zijn rustpositie toegaat. Bij een tableau wordt de laatste waarde vastgehouden zolang er op het tableau sprake is van contact. De paddle houdt de waarde van de actuele positie op het beeldscherm totdat die positie verandert. Die eigenschap kunnen we benutten om met de paddle te tekenen: de opgevraagde waarden gebruiken we dan om punten te zetten (zie programma PDL 2). Maar pas op: de Y-waarde loopt maar tot 192 terwijl bij PDL(2) waarden tot en met 255 worden uitgevoerd.

PDL 1

```
10 COLOR 1,15
20 SCREEN2
30 SPRITE$(1)=STRING$(4,0)+CHR$(255)+
CHR$(255)+CHR$(24)+CHR$(24)
40 PUT SPRITE 1,(PDL(1),180)
50 GOTO 40
```

PDL 2

```
10 COLOR 1,15
20 SCREEN 2
30 PSET(PDL(1),PDL(2))
40 GOTO 30
```

8 Interruptbesturing in machinetaal

8.1 SPRITE ON/OFF/STOP

De instructie SPRITE ON/OFF/STOP hangt direct samen met de BASIC-instructie ON SPRITE GOSUB. Samen zijn ze verantwoordelijk voor de interruptbesturing. Dat wil zeggen dat tijdens de uitvoering van een BASIC-programma om de zoveel tijd wordt gecontroleerd of een bepaalde gebeurtenis wel of niet plaatsvindt. De uitvoering van het programma wordt daarbij niet onderbroken. Bij SPRITE ON wordt er getest of er een botsing tussen sprites plaatsvindt. Als dat het geval is, vertakt het programma naar een subroutine die in het begin van het programma is aangegeven met ON SPRITE GOSUB. De toepassingen van deze sprite-botsingtest moeten we vooral zoeken op spelletjesterrein. Bijvoorbeeld: twee ruimteschepen botsen op elkaar. De subroutine zorgt ervoor dat op die plek een explosie zichtbaar wordt. Of: iemand is bezig rondvliegende objecten neer te schieten. Zodra zijn munitie (een sprite) een object (ook een sprite) raakt, vertakt de computer naar een subroutine waardoor de score omhoog gaat en beide sprites van het toneel verdwijnen. De vertakking naar de subroutine (waarvan het regelnummer in de instructie ON SPRITE GOSUB staat), kunt u op elk gewenst moment tegenhouden met SPRITE OFF, weer toelaten met SPRITE ON of, na SPRITE STOP, veel later opnieuw toelaten met SPRITE ON. Dat laatste kunt u bijvoorbeeld toepassen in een spel tussen een speler en de computer waarin na enige tijd de rollen omdraaien. Op dat moment moeten de botsingen tussen sprites een andere betekenis krijgen. De eerste betekenis schakelt u uit met SPRITE OFF. Als later de rollen weer wisselen, herstelt u de oorspronkelijke toestand met SPRITE ON. Voor de computer is er al sprake van een botsing als 1 afzonderlijke punt van een sprite-matrix een punt van een andere sprite-matrix overlapt. Soms is er dan op het scherm nog geen botsing te zien omdat lang niet elke sprite de hele matrix in beslag neemt. Een lastig geval doet zich voor als u sprites met meer kleuren wilt gebruiken. Een sprite met drie kleuren bestaat eigenlijk uit drie afzonderlijke 1-kleurige sprites die elkaar overlappen en (zo veel mogelijk) synchroon bewegen. Er is met andere woorden voortdurend een drievoudige botsing zodat de computer constant wil vertakken naar een botsingroutine. Er blijft niets anders over dan te kiezen: of meerkleurige sprites of de mogelijkheid om met botseffecten te werken. Als de computer een seintje krijgt over een botsing tussen twee sprites, moet u meteen de botsingtest uitschakelen, anders blijft deze signalen geven tot de sprites elkaar volledig overlappen. Dat is op zichzelf niet zo erg maar het automatische gevolg (vertakking naar

de botsingroutine) belemmert het verdere programmaverloop. De instructie SPRITE ON maakt geen onderscheid tussen sprites die op elkaar botsen. Twee sprites van dezelfde partij kunnen dus best een knal veroorzaken. Om te weten te komen welke sprite de botsing heeft veroorzaakt, zou u in de sprite attribute table kunnen nagaan op welke coördinaten twee sprites elkaar overlappen. Helaas is dat zoekwerk een tijdrovende zaak. Een andere mogelijkheid is gebruik te maken van een aparte variabele waarin het nummer komt te staan van de sprite die het laatst heeft bewogen. De 'passieve' sprite bevindt zich in de matrix van de actieve sprite. U kunt dan met VPEEK in het VRAM snel nagaan om welke sprite het gaat. Het vervolg hangt dan af van de spelregels: punten erbij omdat de passieve sprite werd getroffen of juist punten eraf omdat de actieve sprite zo dom was de botsing te veroorzaken. Tenslotte moet de computer nog uitmaken van welke partij de passieve sprite deel uitmaakt. Het is dan het handigste een bepaald systeem te gebruiken bij de toekenning van spritenummers aan de partijen. Als u dat willekeurig doet, wordt het programmeren er alleen maar ingewikkelder van. Net als bij iedere andere BASIC-instructie wordt de informatie over een sprite-botsing verkregen uit gegevens die al ergens anders in het geheugen zijn opgeslagen. Als u de inhoud van register 8 van de VDP opvraagt met

```
PRINT VDP(8)
```

krijgt u altijd een melding over botsingen tussen sprites, zelfs als u de instructie SPRITE OFF hebt ingevoerd. Bit 5 geeft daarbij aan of er wel (1) of niet (0) sprake is van overlapping. U moet dit register alleen lezen om daarmee op een nuttige onderbreking van het programma te kunnen reageren of een zinloze onderbreking te voorkomen. In andere gevallen wordt het programma er een stuk langzamer door in vergelijking met de interruptbesturing door ON SPRITE of SPRITE ON/OFF. U moet goed in de gaten houden dat wanneer u register 8 van de VDP oproept, de botsingtest automatisch wordt uitgeschakeld (hetzelfde effect als SPRITE OFF).

De programma's STRIGO 1-4 staan aan het einde van deze paragraaf.

MSX-BASIC heeft een bijzondere eigenschap die goed van pas kan komen: ook zonder kennis van machinetaal kunt u profiteren van de snelheid daarvan. Interruptroutines werken in machinetaal, maar u kunt ze indirect in BASIC-programma's toepassen. Dergelijke routines beginnen met het woord ON (ON ERROR bij een fout, ON INTERVAL bij vaste tijdsperiodes, ON KEY als een functietoets wordt ingedrukt, enzovoort). Is een functie eenmaal gedefinieerd, dan voert de MSX die automatisch uit zodra hij op de betreffende ON-instructie stuit. De schakelaar voor de interruptroutines zorgt ervoor dat bij ON de zelfstandige controle door de computer start en dat na OFF de test en het erbij horende resultaat worden uitgeschakeld. Na STOP gaan de regelmatige tests wel door maar het eventuele resultaat (vertakking naar een subroutine) is dan uitgeschakeld. In het laatste geval wordt na STRIG(N) ON alsnog de bijbehorende joystick uitgelezen en zonodig vertakt het programma naar de betreffende routine. De bedoeling daarvan is te voorkomen dat het programma binnen een joystick-routine (na STRIG(N) ON) weer naar dezelfde routine springt; dat zou een eindeloze, recursieve beweging tot gevolg hebben. Pas na de verwerking van een subroutine mag er weer een nieuwe STRIG(N) ON-instructie volgen waarbij de computer tegelijk de informatie verwerkt die in de STRIG(N) STOP- fase is opgeslagen (zie programma STRIGO 1). De instructie STRIG(N) ON/OFF/STOP heeft betrekking op de interruptroutine die nagaat of de spatiebalk of een drukknop van een joystick is ingedrukt. Parameter N heeft de volgende betekenis.

STRIG(0) ON	spatiebalk
STRIG(1) ON	joystick 1, knop 1
STRIG(2) ON	joystick 2, knop 1
STRIG(3) ON	joystick 1, knop 2
STRIG(4) ON	joystick 2, knop 2

De laatste twee mogelijkheden mag u alleen gebruiken als u een echte MSX-joystick bezit; zo'n joystick heeft twee drukknoppen die ieder hun eigen functie hebben. De instructie STRIG(N) ON/OFF/STOP heeft vooral zin als we de joystick gebruiken voor spelletjes. Een paar voorbeelden.

- U wilt niet dat een speler in de eerste ronde van het spel meerdere pijlen na elkaar kan afschieten. Daaarom zorgt u ervoor dat de toestand STRIG(N) ON gehandhaafd blijft tot er een pijl is afgevuurd. Direct daarna wordt STRIG(N) OFF ingevoerd. Vervolgens laat u de pijl zijn baan afmaken tot hij buiten beeld komt. Zodra dat het geval is, treedt

STRIG(N) ON weer op. Pas dan kan opnieuw een pijl worden afgeschoten. In programma STRIGO 2 is deze procedure toegepast. Om het verschil tussen de opties ON, OFF en STOP duidelijk te maken, voert u die achtereenvolgens in in regel 110. Als er een pijl op het scherm verschijnt, drukt u op de spatiebalk. De verschillen zijn groot:

STRIG(N) ON
na een tijdje heeft u niet veel pijlen meer op uw boog want die vliegen allemaal over het scherm; steeds als u de spatiebalk indrukt, vertakt het programma naar de subroutine en verschijnt er een nieuwe pijl

STRIG(N) OFF
er gebeurt niet bijzonders

STRIG(N) STOP
de computer onthoudt de opdracht van de toets/drukknop en voert die uit zodra de vorige pijl van het scherm is verdwenen

- Bij een ander spelletje mag u pas op de drukknop drukken als er een toon klinkt. Programma STRIGO 3 laat zien hoe STRIG(N) ON/OFF/STOP hierbij van dienst kan zijn.
- Twee spelers mogen om de beurt de drukknop van hun joystick gebruiken. Met de parameter N kunnen we ze van elkaar onderscheiden en ervoor zorgen dat zodra speler A zijn drukknop heeft ingedrukt, deze wordt uitgeschakeld en die van speler B tegelijkertijd wordt ingeschakeld. Deze mogelijkheid is verwerkt in programma STRIGO 4.

STRIGO 1

```
10 SCREEN 0
20 PRINT "drukt u gedurende de"
30 PRINT "getalkeuze een keer op de spatiebalk"
40 FOR N=1 TO 1000
50 NEXT N
60 ON STRIG GOSUB 140
70 STRIG(0) ON
80 STRIG(0) STOP
90 FOR N=1 TO 100
100 PRINT N
110 NEXT N
120 STRIG(0) ON
130 GOTO 130
140 PRINT"spatiebalk ingedrukt"
```

STRIGO 2

```
10 SCREEN 0
20 COLOR 1,15
30 PRINT "drukt u om te"
40 PRINT "schieten op de spatiebalk"
50 FOR N=1 TO 1000
60 NEXT N
70 SCREEN 2
80 ON STRIG GOSUB 110
90 STRIG(0) ON
100 GOTO 100
110 STRIG(0) OFF
120 FOR N=191 TO 1 STEP -2
130 DRAW "c1 bm 128,=n; u8 n f4 g4"
140 DRAW "c15 bm 128,=n; u8 n f4 g4"
150 NEXT N
160 STRIG(0) ON
170 RETURN
```

STRIGO 3

```
10 SCREEN 0
20 PRINT "druk op de spatiebalk"
30 PRINT "als de noot C klinkt"
40 PRINT
50 ON STRIG GOSUB 160
60 STRIG(0) ON
70 FOR N=1 TO 10
80 READ A$
90 IF A$="C" THEN STRIG(0) ON ELSE STRIG(0) OFF
100 PLAY "L1"+A$
110 IF PLAY(1) THEN GOTO 110
120 NEXT N
130 PRINT "van 3 kansen ";Z;" goed
140 END
150 DATA C,E,F,C,D,F,A,C,D,E
160 Z=Z+1
```

```
170 STRIG(0) OFF
180 RETURN
```

STRIGO 4

```
10 SCREEN 0
20 COLOR 1,15
30 ON STRIG GOSUB ,240,240
40 PRINT "speler 1 kan nu"
50 PRINT "met joystick 1 schieten"
60 FOR N=1 TO 1000
70 NEXT N
80 STRIG(2) OFF
90 STRIG(1) ON
100 SCREEN 2
110 FOR N=1 TO 1000
120 NEXT N
130 SCREEN 0
140 PRINT "speler 2 kan nu"
150 PRINT "met joystick 2 schieten"
160 FOR N=1 TO 1000
170 NEXT N
180 STRIG(2) ON
190 STRIG(1) OFF
200 SCREEN 2
210 FOR N=1 TO 1000
220 NEXT N
230 RUN
240 STRIG(1) OFF
250 STRIG(2) OFF
260 FOR M=191 TO 0 STEP-2
270 DRAW "c1 bm 128,=m; u16 n f4 g4"
280 DRAW "c15 bm 128,=m; u16 n f4 g4"
290 NEXT M
300 RETURN
```

8.3 ON SPRITE GOSUB

De programma's OSPRI 1 en 2 staan aan het einde van deze paragraaf.

We hebben gezien wat de functie is van instructies die met ON beginnen. Bij de instructie ON SPRITE GOSUB gaat het er om of ergens op het scherm twee sprites elkaar raken. Zo ja, dan moet het programma vertakken naar de regel die achter GOSUB vermeld staat. Ook deze instructie vindt haar toepassing voornamelijk in spelletjes zoals de voorbeelden in deze paragraaf en de spelen Freddie en Mr. Miner in hoofdstuk 9. Programma OSPRI 1 laat zien

hoe de computer onderzoekt op welk moment twee raketten op elkaar botsen. Telkens als dat gebeurt, hoort u kort na elkaar vier pieptonen. De instructies ON SPRITE GOSUB (regel 30) en SPRITE ON (regel 40) geven de computer opdracht te letten op botsingen tussen sprites en in dat geval te vertakken naar regel 120. ON SPRITE schakelt u uit met SPRITE OFF. Na SPRITE STOP gaat de test door, maar de vertakking naar de subroutine blijft achterwege. Dat gebeurt pas op het moment dat SPRITE ON weer is ingevoerd. In feite is dit hetzelfde verhaaltje als in paragraaf 8.1. Om te weten wie op wie botst moeten de bewegingen van de sprites worden opgeslagen in een aparte variabele (zie programma OSPRI 2). De toewijzing van de waarden 1 (voor sprite 1) of 2 (voor sprite 2) aan variabele Z moet plaatsvinden voordat de sprite beweegt want anders verdwijnt de computer naar de subroutine zonder te weten wie de boosdoener is. Daarnaast is het aan te raden de ON SPRITE-routine uit te schakelen zodra het programma de subroutinelus heeft bereikt. Probeer maar eens wat er gebeurt als u de instructie SPRITE OFF weglaat. Bij de eerstvolgende beweging overlappen de sprites elkaar en daarna nog twee keer. Volgt er intussen geen SPRITE OFF-instructie, dan gaat de computer steeds weer naar het begin van de botsingroutine in regel 140 zonder die verder af te handelen. Op het scherm kunt u zien dat het steeds sprite 2 is die de botsing veroorzaakt (regel 110). Nog een belangrijk punt: de instructie ON SPRITE GOSUB wordt geactiveerd zodra ook maar ergens op het scherm sprites elkaar overlappen, ongeacht hoe dat gebeurt. Het kan dus best gebeuren dat twee mannetjes die elkaar (te) dicht naderen er al voor zorgen dat de ON SPRITE-routine wordt uitgevoerd; de sprite-rasters zijn immers meestal groter dan de sprites zelf. Om dat te voorkomen hebben we eigenlijk een krachtige subroutine nodig. Een voorlopige oplossing is de sprites niet puntsgewijs maar stapsgewijs (met meerdere punten tegelijk) te laten bewegen. De beweging is dan niet vloeiend, maar de SPRITE ON-instructie wordt wel nauwkeurig uitgevoerd.

Opmerkingen

- Ten overvloede wijzen we er nog eens op dat u niet moet vergeten de SPRITE ON-instructie tijdig uit te schakelen met SPRITE OFF. U voorkomt daarmee dat het programma voortdurend naar de subroutine vertakt.
- U kunt het register dat verantwoordelijk is voor sprite-botsingen op twee manieren lezen: met de instructie VDP (het gaat om bit 5 van register 8) of met de systeemvariabelen vanaf adres &HF3DF (zie ook de paragraaf over VDP).

OSPRI 1

```
10 COLOR 1,15
20 SCREEN 2
30 ON SPRITE GOSUB 130
40 SPRITE ON
50 SPRITE$(1)="UUUUUUUU"
60 SPRITE$(2)="UUUUUUUU"
70 FOR N=1 TO 255
80 PUT SPRITE 1,(N,96)
90 PUT SPRITE 2,(256-N,96)
100 NEXT N
110 SPRITE ON
120 GOTO 60
130 SPRITE OFF
140 BEEP
150 RETURN
```

OSPRI 2

```
10 COLOR 1,15
20 SCREEN 1
30 ON SPRITE GOSUB 150
40 SPRITE ON
50 SPRITE$(1)="UUUUUUUU"
60 SPRITE$(2)="UUUUUUUU"
70 FOR N=1 TO 255
80 Z=1
90 PUT SPRITE 1,(N,96)
100 Z=2
110 PUT SPRITE 2,(256-N,96)
120 NEXT N
130 SPRITE ON
140 GOTO 60
150 SPRITE OFF
160 PRINT Z;
170 CLOSE
180 SPRITE OFF
190 RETURN
```

8.4 ON STRIG GOSUB

De programma's OSTRI 1 en 2 staan aan het einde van deze paragraaf.

Met de instructie ON STRIG GOSUB gaat de computer na of de drukknop van een joystick of de spatiebalk is ingedrukt (zie programma OSTRI 1). Deze instructie hangt nauw samen met STRIG ON/OFF/STOP (paragraaf 8.2). De verschillende functies daarvan hoeven we hier niet meer te bespreken. Als we regel 30 van programma OSTRI 1 veranderen, kan het programma afhankelijk van de invoer vertakken naar verschillende subroutines. De regelnummers daarvan moeten dan achter elkaar worden ingevoerd, gescheiden door een komma (zie OSTRI 2a). Drukt u op de spatiebalk, dan vertakt het programma naar regel 1000. Wordt drukknop 2 van joystick 2 ingedrukt, dan springt het programma naar regel 5000. Willen we werkelijk alle drukknoppen van de joysticks kunnen controleren, dan is de uitbreiding OSTRI 2b noodzakelijk. Nog een kleine uitbreiding erbij (OSTRI 2c) en u krijgt alles wat er te weten valt op het scherm te zien. U weet dan meteen of de joysticks die u gebruikt wel echt passen bij de MSX. Als het goed is, komen er dan achter elkaar vier meldingen op het scherm:

1 Joystick 1, drukknop 1 ingedrukt
2 Joystick 2, drukknop 1 ingedrukt
3 Joystick 1, drukknop 2 ingedrukt
4 Joystick 2, drukknop 2 ingedrukt

Als het in een spelletje te pas komt dat iedere speler evenveel keer mag drukken, kunt u in het programma een teller inbouwen en de speler die het eind van de reeks heeft bereikt met STRIG(N) OFF van verdere deelname uitsluiten. Experimenteer zelf een beetje met deze instructies in de spelletjes die u maakt. Verder is het leerzaam de twee spelen in hoofdstuk 9, Freddie en Mr. Miner, eens helemaal uit te pluizen.

OSTRI 1

```
10 COLOR 1,15
20 SCREEN 0
30 ON STRIG GOSUB 60
40 STRIG(0) ON
50 GOTO 50
60 PRINT "Spatiebalk ingedrukt"
70 RETURN
```

OSTRI 2a

```
10 COLOR 1,15
20 SCREEN 0
30 ON STRIG GOSUB 1000,2000,3000,4000,5000
```

OSTRI 2b

```
40 FOR N=0 TO 4
50 STRIG(N) ON
60 NEXT N
```

OSTRI 2c

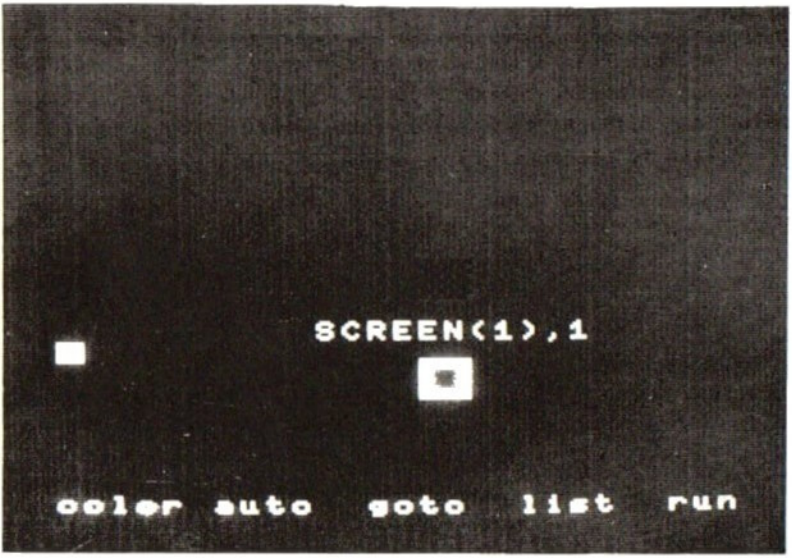
```
999 GOTO 999
1000 PRINT "Spatiebalk ingedrukt"
1010 RETURN
2000 PRINT "Joystick 1, drukknop 1 ingedrukt"
2010 RETURN
3000 PRINT "Joystick 2, drukknop 1 ingedrukt"
3010 RETURN
4000 PRINT "Joystick 1, drukknop 2 ingedrukt"
4010 RETURN
5000 PRINT "Joystick 2, drukknop 2 ingedrukt"
5010 RETURN
```

9.1 Wisseling van sprites

Tengevolge van een SCREEN-instructie wordt het video-RAM gewist, bijvoorbeeld bij een sprong van tekst (SCREEN 0/1) naar grafiek (SCREEN 2/3). Verandering van de tweede SCREEN-parameter (bijvoorbeeld SCREEN,1) wist de sprite generator table. Daarom moeten bij wijziging van de sprite-grootte alle sprite-DATA opnieuw worden ingelezen. Met de instructie VDP kan de sprite-grootte worden veranderd zonder de sprites in het VRAM aan te tasten. Het programma in deze paragraaf geeft hiervan een indrukwekkende demonstratie. Eerst worden 8*8-sprites normaal afgebeeld (1 sprite-punt = 1 beeldpunt: SCREEN,0) en daarna vergroot (1 op 4: SCREEN,1). Ondanks deze wijziging verandert de vorm niet; alleen aan het begin van het programma wordt het sprite-patroon geladen.

```
10 REM Programma 1
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM 8*8-sprites normaal en vergroot zonder VRAM te wissen
50 SCREEN 1
60 COLOR 15,4
70 REM Sprite 1: schaakbordpatroon
80 DATA 01010101
90 DATA 10101010
100 DATA 01010101
110 DATA 10101010
120 DATA 01010101
130 DATA 10101010
140 DATA 01010101
150 DATA 10101010
160 REM Sprite 2: kader
170 DATA 11111111
180 DATA 11111111
190 DATA 11000011
200 DATA 11000011
210 DATA 11000011
220 DATA 11000011
230 DATA 11111111
240 DATA 11111111
250 REM Sprites 0 en 1 inlezen
```

```
260 RESTORE 80
270 FOR M= 0 TO 1
280 B$=""
290 FOR N=0 TO 7
300 READ A$
310 REM Binaire representatie van karakters veranderen
320 B$=B$+CHR$(VAL("&B"+A$))
330 NEXT N
340 SPRITE$(M)=B$
350 NEXT M
360 FOR A=1 TO 100
370 NEXT A
380 REM Afbeelding van sprite 0 en 1
390 PUT SPRITE 0,(128,86),1,0
400 PUT SPRITE 1,(128,126),15,1
410 REM Test van bit 0 in het VDP-register 1
420 B=VDP(1) AND 1
430 REM Als bit 0 = 1, dan is nu bit 0 = 0
440 IF B=1 THEN VDP(1)=VDP(1) AND &B11111110:
LOCATE 10,14:PRINT "SCREEN(1),0":GOTO 360
450 REM Als bit 0 = 0,dan is nu bit 0 = 1
460 IF B=0 THEN VDP(1)=VDP(1) OR &B00000001:
LOCATE 10,14:PRINT "SCREEN(1),1":GOTO 360
```

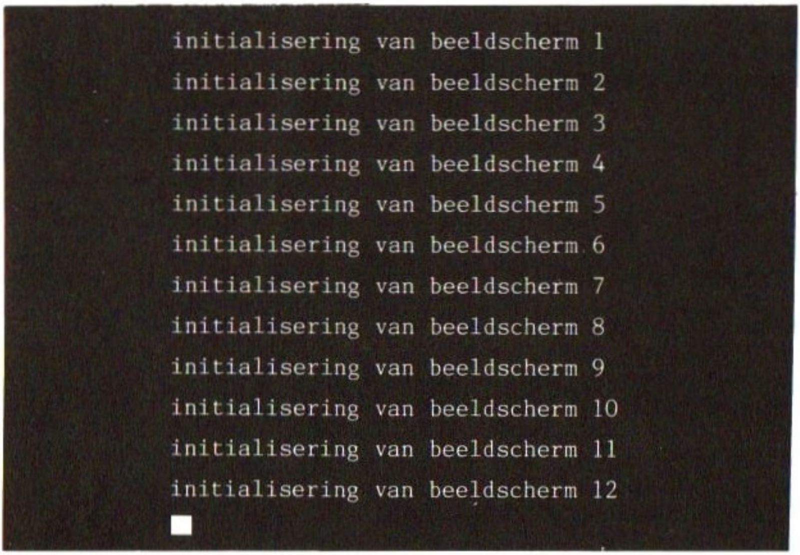


9.2 13 beeldschermen in SCREEN 0

Het maximale aantal tekens op het scherm (en dus in het VRAM) bedraagt in SCREEN 0 960. Die nemen samen 2048 van de in totaal 16.000 VRAM-adressen in beslag. Het resterende deel kan worden gebruikt voor de opslag van variabelen (met VPEEK en VPOKE) of sprites (met SPRITE$), wat niet aan te bevelen is. Het onderstaande programma maakt beter gebruik van het VRAM; de geheugenruimte wordt in beslag genomen door verschillende text screens. De instructie VDP maakt niet alleen de opslag daarvan mogelijk, maar zorgt ook voor een snelle afbeelding op het scherm. Op deze manier kunt u bijvoorbeeld hulpinformatie opslaan of formules die u vaak nodig heeft. Dankzij de supersnelle wisseling van de beeldschermen kunt u uw fantasie de vrije loop laten. Zelfs een eenvoudige tekenfilm behoort tot de mogelijkheden.

```
10 REM Programma 2
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM 13 beeldschermen in SCREEN 0
50 SCREEN 0
60 WIDTH 39
70 KEY OFF
80 REM Na onderbreking van het programma altijd naar het
 invoerscherm terugkeren
90 ON STOP GOSUB 430
100 STOP ON
110 M1=200
120 REM De beeldschermen 1 t/m 12 worden voorzien van informatie
130 FOR N=1 TO 12
140 PRINT "Initialisering van beeldscherm";N
150 A$="Dit is beeldscherm"+STR$(N)+STRING$(10,32)
160 REM De informatie (A$) wordt op de beginpositie van de 12
 beeldschermen gePOKEt
170 FOR M=1 TO LEN(A$)
180 VPOKE (3+N)*1024+M,ASC(MID$(A$,M,1))
190 NEXT M
200 NEXT N
210 CLS
220 PRINT"Dit is beeldscherm 0"
230 VDP(2)=0
240 REM Naar vertragingslus, indien gewenst
250 GOSUB 390
260 REM Beeldschermen 1 t/m 12 doorbladeren
270 FOR N=4 TO 15
```

```
280 VDP(2)=N
290 REM Naar vertragingslus, indien gewenst
300 GOSUB 390
310 NEXT N
320 VDP(2)=0
330 PRINT
340 B$=""
350 INPUT "Wilt u de snelheid invoeren          (1000 TOT 1) ";B$
360 M1=VAL(B$)
370 GOTO 230
380 REM Vertragingslus
390 FOR M=1 TO M1
400 NEXT M
410 RETURN
420 REM Terug naar invoerscherm; programma beëindigen door
 CTRL en STOP tegelijkertijd in te drukken
430 VDP(2)=0
440 END
```



9.3 Twee karaktersets

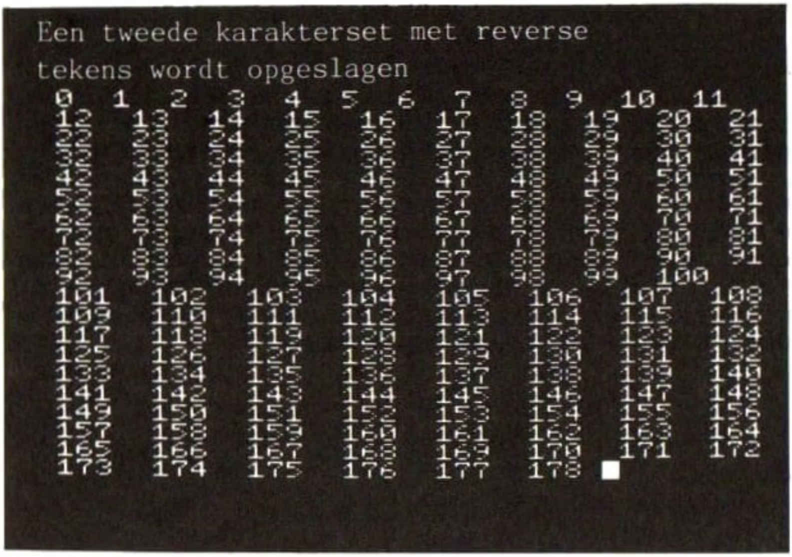
Op de text screens SCREEN 0 en 1 is plaats voor maximaal 256 tekens tegelijkertijd. De volgende BASIC-regels zetten het zichtbare gedeelte daarvan op het scherm:

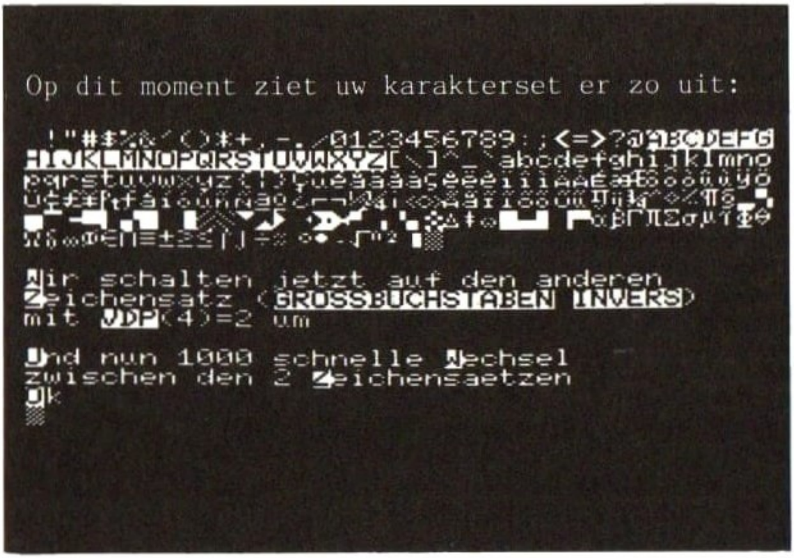
```
10 FOR N=32 TO 255
20 PRINT CHR$(N);
30 NEXT N
```

Wijziging van de vorm van een karakter heeft tot gevolg dat de karakters op het scherm met dezelfde tekencode ook veranderen (zie paragraaf 4.1 en 4.2). Het programma in deze paragraaf is helaas ook niet bij machte meer dan 256 tekens tegelijkertijd weer te geven. Daar staat tegenover dat het de twee karaktersets niet alleen in het VRAM opslaat, maar ook de mogelijkheid biedt deze in een fractie van een seconde te verwisselen. De instructie VDP bewijst daarbij weer eens zijn nut. Om het programma kort te houden is de tweede karakterset gelijk aan de standaard karakterset, met dien verstande dat de grote letters invers zijn weergegeven.

```
10 REM Programma 3
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM Twee verschillende karaktersets; VDP(4) verwisselt ze
50 SCREEN 0
60 WIDTH 40
70 PRINT "De tweede karakterset met inverse hoofdletters wordt in
 het geheugen opgeslagen"
80 PRINT
90 REM De karakters 0 t/m 64 worden in dezelfde vorm in de tweede
 karakterset opgenomen
100 FOR N=0 TO 64
110 PRINT N;
120 FOR M=1 TO 8
130 GOSUB 660
140 NEXT M
150 NEXT N
160 REM De karakters 65 t/m 90 (hoofdletters) worden voor de
 tweede karakterset omgerekend en daardoor invers weergegeven
170 FOR N=65 TO 90
180 PRINT N;
190 FOR M=1 TO 8
200 A$=BIN$(VPEEK(2047+(N*8)+M))
210 A$=STRING$(8-LEN(A$),"0")+A$
220 REM De binaire representatie omdraaien: 0 wordt 1 en
 omgekeerd
230 FOR L=1 TO 8
```

```
240 IF MID$(A$,L,1)="0" THEN MID$(A$,L,1)="1" ELSE MID$(A$,L,1)="0"
250 NEXT L
260 VPOKE 4095+(N*8)+M,VAL("&B"+A$)
270 NEXT M
280 NEXT N
290 REM De karakters 91 t/m 254 komen precies zo in karakterset 2
 te staan
300 FOR N=91 TO 254
310 PRINT N;
320 FOR M=1 TO 8
330 GOSUB 660
340 NEXT M
350 NEXT N
360 REM Cursor in de vorm van schaakbord
370 FOR M=1 TO 8 STEP 2
380 VPOKE 4095+(255*8)+M,85
390 VPOKE 4095+(255*8)+M+1,170
400 NEXT M
410 CLS
420 PRINT"Op dit moment ziet uw karakterset er zo uit:"
430 PRINT
440 REM Weergave van de op het beeldscherm zichtbare karakterset
450 FOR N=32 TO 255
460 PRINT CHR$(N);
470 NEXT N
480 PRINT
490 PRINT
500 GOSUB 690
510 PRINT "En dan nu de volgende karakterset dank zij VDP(4)=2"
520 PRINT
530 GOSUB 690
540 REM Over naar karakterset 2
550 VDP(4)=2
560 PRINT"1000 maal van de een naar de ander"
570 GOSUB 690
580 FOR N=1 TO 1000
590 REM Karakterset 1 oproepen
600 VDP(4)=1
610 REM Karakterset 2 oproepen
620 VDP(4)=2
630 NEXT N
640 END
650 REM Karakterset kopiëren
660 VPOKE 4095+(N*8)+M,VPEEK(2047+(N*8)+M)
670 RETURN
680 REM Wachtlus
690 FOR N=1 TO 1000
700 NEXT N
710 RETURN
```

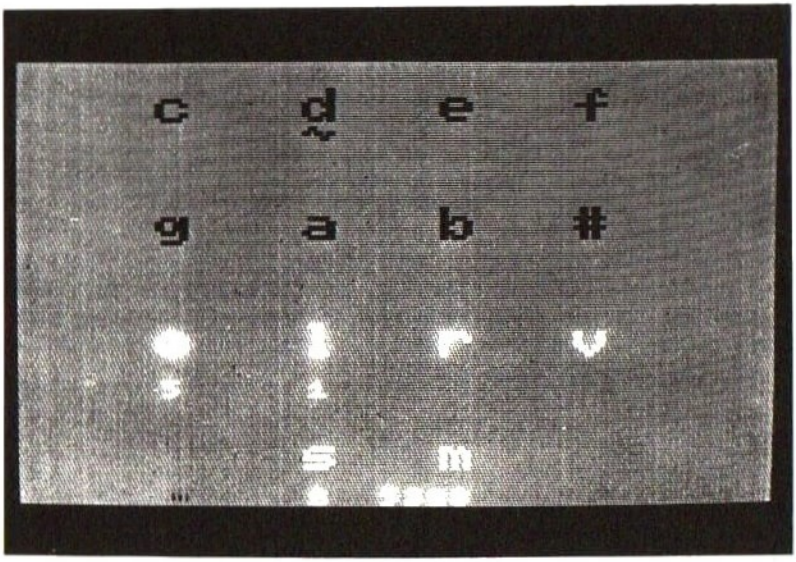

Een tweede karakterset met reverse
tekens wordt opgeslagen
0 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41
42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61
62 63 64 65 66 67 68 69 70 71
72 73 74 75 76 77 78 79 80 81
82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100
101 102 103 104 105 106 107 108
109 110 111 112 113 114 115 116
117 118 119 120 121 122 123 124
125 126 127 128 129 130 131 132
133 134 135 136 137 138 139 140
141 142 143 144 145 146 147 148
149 150 151 152 153 154 155 156
157 158 159 160 161 162 163 164
165 166 167 168 169 170 171 172
173 174 175 176 177 178


Op dit moment ziet uw karakterset er zo uit:
Wir schalten jetzt auf den anderen
Zeichensatz (GROSSBUCHSTABEN INVERS)
mit VDP(4)=2 u.m
Und nun 1000 schnelle Wechsel
zwischen den 2 Zeichensaetzen
Ok

9.4 Hoofdletters in sprite-vorm

Zoals u weet kunt u 8*8- of 16*16-sprites normaal of vergroot op het scherm zetten. In het VRAM is plaats voor 256 8*8- of 64 16*16-sprites. Van beide aantallen passen er ten hoogste 32 op het scherm. Daarbij moet u met het volgende rekening houden: op een beeldlijn kunnen maar 4 sprites tegelijkertijd worden weergegeven. Doet u dat niet dan verdwijnt er onherroepelijk wat, stukje bij beetje of hele sprites in een keer. De sprite met het hoogste nummer is het eerst aan de beurt. Het volgende programma laat niet alleen zien hoe u 32 sprites zonder overlapping op het scherm kunt krijgen, maar ook hoe met de instructie VDP de sprites overeen kunnen komen met de gewone karakterset.

```
10 REM Programma 4
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM 32 hoofdletters uit de standaard karakterset in
 sprite-vorm
50 REM 8*8-sprites, vergroot afgebeeld
60 SCREEN 1,1
70 KEY OFF
80 COLOR 1,15
90 REM De vorm van de sprites (VDP(6)) is afhankelijk van die
 van de karakters (VDP(4))
100 VDP(6)=VDP(4)
110 M1=0
120 REM 32 sprites overzichtelijk op het beeldscherm zetten
130 FOR N=32 TO 152 STEP 16
140 FOR M=50 TO 200 STEP 50
150 REM Alleen de sprites in hoofdletters worden afgebeeld
 (a=65); iedere sprite krijgt een kleur toegekend
160 PUT SPRITE M1,(M,N),INT(RND(1)*14)+1,M1+65
170 M1=M1+1
180 NEXT M
190 NEXT N
200 REM Alle 32 sprites gelijktijdig pixelgewijs scrollen
210 FOR L=1 TO 255
220 M1=0
230 FOR N=32 TO 152 STEP 16
240 FOR M=50 TO 200 STEP 50
250 PUT SPRITE M1,(M+L,N),,M1+65
260 M1=M1+1
270 NEXT M
280 NEXT N
290 NEXT L
```

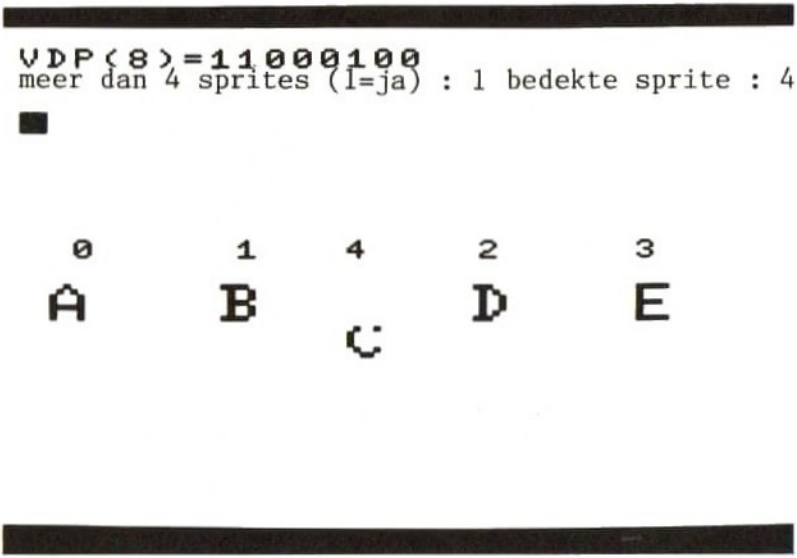

c d e f
g a b #

9.5 4 sprites per regel

Sprites van 8*8 of 16*16 kunnen hun normale vorm hebben of tweemaal zo groot zijn. In het VRAM is plaats voor 256 verschillende sprites van 8*8 of 64 sprites van 16*16. Van beide aantallen passen er ten hoogste 32 op het scherm. Daarbij moet u met het volgende rekening houden: op een beeldlijn kunnen maar 4 sprites tegelijkertijd worden weergegeven. Doet u dat niet dan verdwijnt er onherroepelijk wat, stukje bij beetje of hele sprites in een keer. De sprite met het hoogste nummer is het eerst aan de beurt. Het programma in deze paragraaf demonstreert met een aantal voorbeelden hoe de overlapping op het scherm in zijn werk gaat, dat wil zeggen welke sprite overlapt wordt als er meer dan 4 sprites op een horizontale lijn staan.

```
10 REM Programma 5
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM Demonstratieprogramma: horizontaal passen maar 4 sprites
 gelijktijdig op het scherm; de sprite met het hoogste getal wordt
 stukje bij beetje gewist
50 REM 8*8-sprites, vergroot afgebeeld
60 SCREEN 1,1
70 KEY OFF
80 COLOR 1,15
90 WIDTH 29
100 REM De vorm van de sprites (VDP(6)) is afhankelijk van die
 van de karakters (VDP(4))
110 VDP(6)=VDP(4)
120 REM 3 toestanden worden getest
130 FOR L=1 TO 3
140 READ N0,N1,N2,N3,N4
150 REM De sprites die niet veranderen
160 PUT SPRITE N0,(25,96),,65
170 PUT SPRITE N1,(75,96),,66
180 PUT SPRITE N3,(150,96),,68
190 PUT SPRITE N4,(200,96),,69
200 LOCATE 1,10
210 REM De sprite-getallen
220 PRINT N0;TAB(7);N1;TAB(11);N2;TAB(16);N3;TAB(22);N4
230 REM Sprite 5 botst op een ander
240 FOR N=70 TO 120
250 PUT SPRITE N2,(112,N),,67
260 REM Inhoud van VDP(8)
270 A$=BIN$(VDP(8))
280 LOCATE 0,1
290 PRINT"VDP(8)=";A$
```

```
300 PRINT"Meer dan 4 sprites (1=ja):";MID$(A$,2,1)
310 PRINT" Bedekte sprite:";:IF MID$(A$,4,5)="11111" THEN PRINT
 "Geen" ELSE PRINT VAL("&b"+MID$(A$,4,5));"    "
320 REM Wachtlus
330 FOR M=1 TO 200
340 NEXT M
350 NEXT N
360 NEXT L
370 LOCATE 0,0
380 END
390 REM DATA-regels voor 3 toestanden
400 DATA 0,1,4,2,3
410 DATA 0,1,2,3,4
420 DATA 4,3,2,1,0
```



9.6 Karaktersets in DATA-regels

In dit boek staan een aantal manieren om de vorm van karakters te veranderen. Door de computer uit te zetten wordt echter alles in het RAM gewist. Wisseling van SCREEN heeft hetzelfde effect. Het programma hieronder biedt de mogelijkheid om een karakterset op band te zetten, vooropgesteld dat de wijzigingen op SCREEN 1 plaatsvonden. De DATA-regels hebben het voordeel dat de karakterset altijd met MERGE aan een ander programma kan worden toegevoegd.



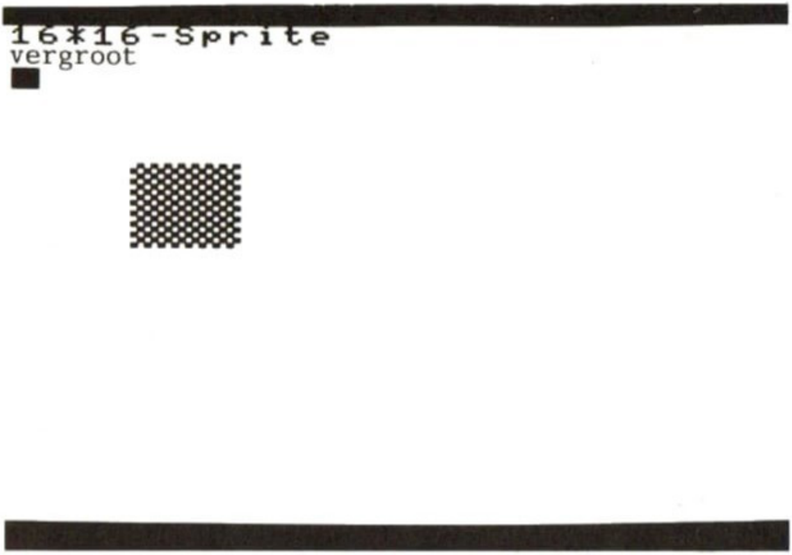

```
10 REM Programma 6
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM Programma om karakterset uit geheugen (SCREEN 1) op band
 te zetten; de DATA-regels kunnen m.b.v. een laadroutine in een
 ander programma worden geMERGEd
50 CLS
60 KEY OFF
70 REM SCREEN 1 oproepen
80 WIDTH 29
90 PRINT"Karakterset opslaan"
100 PRINT"-------------------"
110 REM Karakterset begrenzen
120 INPUT "Beginwaarde ";M1
130 INPUT "Eindwaarde ";M2
```

```
140 IF M1>M2 OR M2>255 OR M1<0 THEN RUN
150 PRINT
160 REM Gekozen karakter
170 PRINT "Zo zien de karakters van uw keuze er uit:"
180 PRINT
190 FOR N=M1 TO M2
200 PRINT CHR$(N);
210 NEXT N
220 PRINT
230 PRINT
240 INPUT "Is dit zo goed (j/n)";F$
250 IF F$="J" OR F$="j" THEN GOTO 260 ELSE RUN
260 F$=""
270 PRINT
280 INPUT "Naam karakterset ";F$
290 PRINT
300 PRINT "Recorder op opname"
310 PRINT
320 INPUT "Klaar, druk op RETURN";F1$
330 F$="cas:"+F$
340 OPEN F$ FOR OUTPUT AS #1
350 REM Zet beginregel op 1000
360 Z=1000
370 REM Laadroutine wordt gemaakt en in geheugen gezet
380 A$=STR$(Z)+" FOR N="+STR$(M1)+" TO "+STR$(M2)
390 GOSUB 850
400 A$=STR$(Z)+" FOR M=0 TO 7"
410 GOSUB 850
420 A$=STR$(Z)+" READ A$"
430 GOSUB 850
440 A$=STR$(Z)+" VPOKE N*8+M"+CHR$(44)+"VAL
("+CHR$(34)+"&B"+CHR$(34)+"+A$)"
450 GOSUB 850
460 A$=STR$(Z)+" NEXT M"
470 GOSUB 850
480 A$=STR$(Z)+" NEXT N"
490 PRINT #1,A$
500 REM Vorm van tekens binair in geheugen opslaan
510 FOR N=M1 TO M2
520 REM De karakters en programmaregels uit het geheugen
530 LOCATE 16,CSRLIN-1
540 PRINT "karakter";N
550 LOCATE 16,CSRLIN
560 PRINT "regel";
570 PRINT USING "####";Z+10;
580 Z=Z+10
590 REM Programmaregel met regelnummer maken
600 A$=STR$(Z)+" REM Karakter "+STR$(N)
610 PRINT #1,A$
```

```
620 FOR M=0 TO 7
630 A$=""
640 Z=Z+10
650 A$=STR$(Z)+" DATA "
660 REM Karakters uit geheugen lezen en vertalen naar binaire
representatie
670 A1$=BIN$(VPEEK(N*8+M))
680 A1$=STRING$(8-LEN(A1$),"0")+A1$
690 A$=A$+A1$
700 REM De volledige regel Z opslaan
710 PRINT #1,A$
720 NEXT M
730 NEXT N
740 CLOSE
750 END
760 REM Testroutine om de gegevens op band te bekijken
770 CLS
780 OPEN "cas:"FOR INPUT AS #1
790 LINE INPUT #1,A$
800 PRINT A$
810 REM File afsluiten wanneer er geen DATA meer op band staan
820 IF EOF(1)=-1 THEN CLOSE:END
830 GOTO 790
840 REM Programmaregels opslaan en regelnummer Z met 10 verhogen
850 PRINT #1,A$
860 Z=Z+10
870 RETURN
```

9.7 Sprite-wisseling 2

Sprites van 8*8 of 16*16 kunnen hun normale vorm hebben of tweemaal zo groot zijn. In het VRAM is plaats voor 256 verschillende sprites van 8*8 of 64 sprites van 16*16. Van beide aantallen passen er ten hoogste 32 op het scherm. Daarbij moet u met het volgende rekening houden: op een beeldlijn kunnen maar 4 sprites tegelijkertijd worden weergegeven. Doet u dat niet dan verdwijnt er onherroepelijk wat, stukje bij beetje of hele sprites in een keer. De sprite met het hoogste nummer is het eerst aan de beurt. Met het volgende programma kunt u na invoer van het sprite-patroon heen en weer schakelen tussen de genoemde formaten. Dat gebeurt met de instructie VDP. In tegenstelling tot de instructie SCREEN laat VDP alle informatie in het VRAM intact.




```
10 REM Programma 7
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM Sprite-weergave 8*8 of 16*16, normaal of tweemaal zo
 groot; SCREEN-instructie en herdefinitie bij omschakeling
 overbodig
50 SCREEN 1
60 KEY OFF
70 COLOR 1,15
80 REM Sprite inlezen (16*16 punten)
90 FOR N=1 TO 16
100 READ A$
110 REM Eerst de linkerkant inlezen (sprite 1 en 2: 8*8)
120 A1$=A1$+CHR$(VAL("&B"+LEFT$(A$,8)))
130 NEXT N
140 RESTORE
150 FOR N=1 TO 16
160 READ A$
170 REM De rechterkant inlezen (sprite 3 en 4: 8*8)
180 A1$=A1$+CHR$(VAL("&B"+RIGHT$(A$,8)))
190 NEXT N
200 REM Sprite-patroon toewijzen aan sprite pattern table
210 FOR N=14336 TO 14367
220 VPOKE N,ASC(MID$(A1$,N-14335,1))
230 NEXT N
240 REM Een grote schaakbord-sprite
```

```
250 DATA 0101010101010101
260 DATA 1010101010101010
270 DATA 0101010101010101
280 DATA 1010101010101010
290 DATA 0101010101010101
300 DATA 1010101010101010
310 DATA 0101010101010101
320 DATA 1010101010101010
330 DATA 0101010101010101
340 DATA 1010101010101010
350 DATA 0101010101010101
360 DATA 1010101010101010
370 DATA 0101010101010101
380 DATA 1010101010101010
390 DATA 0101010101010101
400 DATA 1010101010101010
410 CLS
420 LOCATE 0,0
430 PRINT "8*8-sprite"
440 PRINT "Niet vergroot"
450 REM Bit 0 en 1 uit register 1 worden gereset: 8*8-sprite op
 normale grootte
460 VDP(1)=VDP(1) AND &HFC
470 GOSUB 740
480 CLS
490 LOCATE 0,0
500 PRINT "8*8-sprite"
510 PRINT "Vergroot"
520 REM Bit 0 uit register 1 wordt gezet; de 8*8-sprite wordt
 tweemaal zo groot
530 VDP(1)=VDP(1) OR 1
540 GOSUB 740
550 CLS
560 REM Bit 1 en 0 uit register 1 worden gereset: 8*8-sprite op
 normale grootte
570 VDP(1)=VDP(1) AND &HFC
580 LOCATE 0,0
590 PRINT "16*16-sprite"
600 PRINT "Niet vergroot"
610 REM Bit 1 uit register 1 wordt gezet: 16*16-sprite op normale
 grootte
620 VDP(1)=VDP(1) OR 2
630 GOSUB 740
640 CLS
650 LOCATE 0,0
660 PRINT "16*16-sprite"
670 PRINT "vergroot"
680 REM Bit 0 uit register 1 wordt gezet: 16*16-sprite tweemaal
 zo groot
```

```
690 VDP(1)=VDP(1) OR 1
700 GOSUB 740
710 REM Demonstratie nog een keer; sprite hoeft niet opnieuw te
 worden geladen
720 GOTO 410
730 REM Subroutine voor diagonale sprite-beweging
740 FOR N=1 TO 191
750 PUT SPRITE 0,(N,N),1,0
760 FOR M=1 TO 10
770 NEXT M
780 NEXT N
790 RETURN
```

9.8 Computermuziek met gewone pianotoetsen

De geluids-chip van de MSX stelt u niet alleen in de gelegenheid losse tonen te produceren, maar ook om ze tegelijkertijd over drie kanalen te laten klinken. U kunt daarbij kiezen uit twee mogelijkheden: directe aansturing van de chip-registers of gebruik maken van de MSX-BASIC-instructie PLAY. Hieraan is een nadeel verbonden. Als toetsenist van formaat oriënteert u zich normaal gesproken natuurlijk op de toetsen van uw instrument, maar die faciliteit biedt de MSX niet. De instructie PLAY verlangt de namen van de tonen en SOUND getallen. Het programma in deze paragraaf is daarom een geweldige uitkomst. Het stelt u in staat het toetsenbord van de MSX te gebruiken als gewone toetsen. Een lichte aanslag is voldoende om aan uw computer muzikale hoogstandjes te ontlokken. Zodra het programma is geRUNd wordt op aanschouwelijke wijze duidelijk gemaakt van welke toetsen u gebruik kunt maken. De toon die u speelt, verschijnt bovendien direct op een notenbalk. Composities moeizaam uitschrijven behoort tot het verleden. De functietoetsen 1-5 dienen om drieklanken op te roepen. Die zijn van tevoren via het toetsenbord ingevoerd door een akkoordveld te wissen (SHIFT en de juiste functietoets tegelijk indrukken) en daarna met F6 opnieuw te definiëren.

```
10 REM Programma 8
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM Muziek met een bereik van anderhalf octaaf via het
 toetsenbord van de MSX. Gewone pianotoetsen, notenbalk en
 akkoorden verschijnen op het scherm.
50 REM Met de functietoetsen kunt u akkoorden maken en spelen.
60 COLOR 15,4,4
```

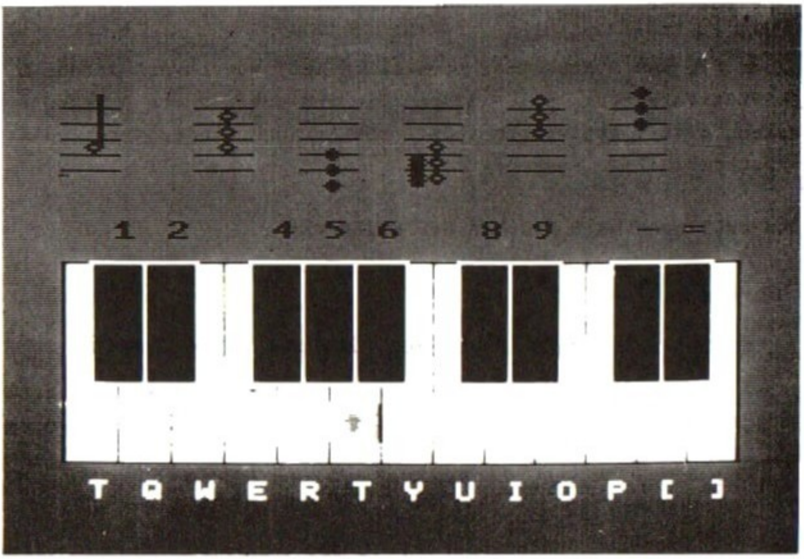
```
70 ON KEY GOSUB 1280,1300,1320,1340,1360,1390,1420,1450,1480,
1510
80 FOR N=1 TO 10
90 KEY(N) ON
100 KEY N,""
110 NEXT N
120 REM Een vergrote 8*8-sprite in de vorm van een pijl geeft aan
 welke toets het laatst is gebruikt
130 SCREEN 2,1
140 REM Witte pianotoetsen tekenen
150 FOR N=24 TO 224 STEP 16
160 LINE (N,80)-(N+16,160),15,BF
170 LINE (N,80)-(N+16,160),1,B
180 NEXT N
190 REM Zwarte pianotoetsen tekenen
200 FOR N=1 TO 9
210 READ A
220 LINE (A,80)-(A+16,128),1,BF
230 LINE (A,80)-(A+16,128),15,B
240 NEXT N
250 OPEN "grp:"FOR OUTPUT AS #1
260 COLOR 15
270 REM Aangeven welke toetsen op het computertoetsenbord
 overeenkomen met welke witte pianotoetsen
280 FOR N=2 TO 14
290 READ A$
300 A=N*16
310 DRAW "bm =a;,168"
320 PRINT #1,A$
330 NEXT N
340 COLOR 1
350 REM Aangeven welke toetsen op het computertoetsenbord
 overeenkomen met welke zwarte pianotoetsen
360 FOR N=1 TO 9
370 READ A,A$
380 DRAW "bm =a;,64"
390 PRINT #1,A$
400 NEXT N
410 FOR N=1 TO 8
420 REM Sprite-pijl inlezen
430 READ A$
440 A1$=A1$+CHR$(VAL("&b"+A$))
450 NEXT N
460 SPRITE$(1)=A1$
470 REM Notenbalk tekenen
480 M1=24
490 GOSUB 1230
500 M1=64
510 GOSUB 1230
```

```
520 M1=96
530 GOSUB 1230
540 M1=128
550 GOSUB 1230
560 M1=160
570 GOSUB 1230
580 M1=192
590 GOSUB 1230
600 REM Controle: is er een toets ingedrukt (j/n)
610 A$=INKEY$
620 IF A$="" THEN 610
630 REM Witte toetsen komen overeen met eerste letterrij van
 toetsenbord (q-])
640 IF ASC(A$)=9 THEN PUT SPRITE 1,(24,135),4,1:
A=50:B=4:C=0:N$="O4C":GOSUB 1030
650 IF ASC(A$)=113 THEN PUT SPRITE 1,(40,135),4,1:
A=47:B=0:C=0:N$="O4D":GOSUB 1030
660 IF ASC(A$)=119 THEN PUT SPRITE 1,(56,135),4,1:
A=44:B=0:C=0:N$="O4E":GOSUB 1030
670 IF ASC(A$)=101 THEN PUT SPRITE 1,(72,135),4,1:
A=41:B=0:C=0:N$="O4F":GOSUB 1030
680 IF ASC(A$)=114 THEN PUT SPRITE 1,(88,135),4,1:
A=38:B=0:C=0:N$="O4G":GOSUB 1030
690 IF ASC(A$)=116 THEN PUT SPRITE 1,(104,135),4,1:
A=35:B=0:C=0:N$="O4A":GOSUB 1030
700 IF ASC(A$)=121 THEN PUT SPRITE 1,(120,135),4,1:
A=32:B=0:C=0:N$="O4B":GOSUB 1130
710 IF ASC(A$)=117 THEN PUT SPRITE 1,(136,135),4,1:
A=29:B=0:C=0:N$="O5C":GOSUB 1130
720 IF ASC(A$)=105 THEN PUT SPRITE 1,(152,135),4,1:
A=26:B=0:C=0:N$="O5D":GOSUB 1130
730 IF ASC(A$)=111 THEN PUT SPRITE 1,(168,135),4,1:
A=23:B=0:C=0:N$="O5E":GOSUB 1130
740 IF ASC(A$)=112 THEN PUT SPRITE 1,(184,135),4,1:
A=20:B=0:C=0:N$="O5F":GOSUB 1130
750 IF ASC(A$)=91 THEN PUT SPRITE 1,(200,135),4,1:
A=17:B=0:C=0:N$="O5G":GOSUB 1130
760 IF ASC(A$)=93 THEN PUT SPRITE 1,(216,135),4,1:
A=14:B=4:C=0:N$="O5A":GOSUB 1130
770 REM Zwarte toetsen komen overeen met cijferrij van
 toetsenbord (1-=), m.u.v. 3, 7 en 0. Witte en zwarte
 toetsen bij elkaar vormen op het computertoetsenbord
 een normale pianotoetsen-configuratie.
780 IF ASC(A$)=49 THEN PUT SPRITE 1,(32,100),4,1:
A=50:B=4:C=47:N$="O4C#":GOSUB 1030
790 IF ASC(A$)=50 THEN PUT SPRITE 1,(48,100),4,1:
A=47:B=0:C=44:N$="O4D#":GOSUB 1030
800 IF ASC(A$)=52 THEN PUT SPRITE 1,(80,100),4,1:
A=41:B=0:C=38:N$="O4F#":GOSUB 1030
```

```
810 IF ASC(A$)=53 THEN PUT SPRITE 1,(96,100),4,1:
A=38:B=0:C=35:N$="O4G#":GOSUB 1030
820 IF ASC(A$)=54 THEN PUT SPRITE 1,(112,100),4,1:
A=35:B=0:C=32:N$="O4A#":GOSUB 1030
830 IF ASC(A$)=56 THEN PUT SPRITE 1,(144,100),4,1:
A=29:B=0:C=26:N$="O5C#":GOSUB 1130
840 IF ASC(A$)=57 THEN PUT SPRITE 1,(160,100),4,1:
A=26:B=0:C=23:N$="O5D#":GOSUB 1130
850 IF ASC(A$)=45 THEN PUT SPRITE 1,(192,100),4,1:
A=20:B=0:C=17:N$="O5F#":GOSUB 1130
860 IF ASC(A$)=61 THEN PUT SPRITE 1,(208,100),4,1:
A=17:B=0:C=14:N$="O5G#":GOSUB 1130
870 GOTO 610
880 REM MSX-toetsen
890 DATA 32,48,80,96,112,144,160,192,208
900 DATA T,Q,W,E,R,T,Y,U,I,O,P,[,]
910 DATA 40,1,56,2,88,4,104,5,120,6,152,8,168,9,200,-,216,=
920 REM Pijl-sprite
930 DATA 11111111
940 DATA 10000001
950 DATA 10011001
960 DATA 10111101
970 DATA 10011001
980 DATA 10011001
990 DATA 10000001
1000 DATA 11111111
1010 GOTO 1010
1020 REM Invoer van akkoord (AK) j/n. Deze optie onbenut laten
 betekent invoer van een losse toon. Op de notenbalk is de stok
 van de noot naar boven gericht.
1030 IF AK>0 THEN GOTO 1550
1040 LINE(24,10)-(40,54),4,BF
1050 PLAY N$
1060 M1=24
1070 GOSUB 1230
1080 DRAW "bm 35,=a;h2g2f2e2nl=b;u20"
1090 REM Zwarte toets betekent #
1100 IF C<>0 THEN DRAW "bm25,=c;":PRINT #1,"#"
1110 RETURN
1120 REM Invoer van akkoord (AK) (j/n). Deze optie onbenut laten
 betekent invoer van een losse toon. Op de notenbalk is het
 stokje van de noot naar beneden gericht.
1130 IF AK>0 THEN GOTO 1550
1140 LINE(24,10)-(40,54),4,BF
1150 PLAY N$
1160 M1=24
1170 GOSUB 1230
1180 DRAW "bm 35,=a;e2f2g2h2nr=b;d20"
1190 REM Zwarte toets betekent #
```

```
1200 IF C<>0 THEN DRAW "bm28,=c;":PRINT #1,"#"
1210 RETURN
1220 REM Notenbalk tekenen
1230 FOR N=20 TO 44 STEP 6
1240 LINE (M1,N)-(M1+16,N),1
1250 NEXT N
1260 RETURN
1270 REM Subroutine voor F1-F5; het akkoord wordt gespeeld
1280 PLAY A$(1),A$(2),A$(3)
1290 RETURN
1300 PLAY B$(1),B$(2),B$(3)
1310 RETURN
1320 PLAY C$(1),C$(2),C$(3)
1330 RETURN
1340 PLAY D$(1),D$(2),D$(3)
1350 RETURN
1360 PLAY E$(1),E$(2),E$(3)
1370 RETURN
1380 REM Akkoorden 1-5 worden ingelezen, laatste akkoord wordt
 gewist
1390 AK=75:M1=64
1400 GOSUB 1680:GOSUB 1230
1410 RETURN
1420 AK=107:M1=96
1430 GOSUB 1680:GOSUB 1230
1440 RETURN
1450 AK=139:M1=128
1460 GOSUB 1680:GOSUB 1230
1470 RETURN
1480 AK=171:M1=160
1490 GOSUB 1680:GOSUB 1230
1500 RETURN
1510 AK=203:M1=192
1520 GOSUB 1680:GOSUB 1230
1530 RETURN
1540 REM De 3 noten per akkoord worden op de notenbalk afgebeeld
 en gespeeld
1550 ZZ=ZZ+1
1560 PLAY N$
1570 DRAW "bm=ak;,=a;c1h2g2f2e2l=b;"
1580 REM Zwarte toets betekent #
1590 IF C<>0 THEN TT=AK-10:DRAW "bm=tt;,=c;":PRINT #1,"#"
1600 IF AK=75 THEN A$(ZZ)=N$
1610 IF AK=107 THEN B$(ZZ)=N$
1620 IF AK=139 THEN C$(ZZ)=N$
1630 IF AK=171 THEN D$(ZZ)=N$
1640 IF AK=203 THEN E$(ZZ)=N$
1650 IF ZZ=3 THEN ZZ=0:AK=0
1660 RETURN
```

```
1670 REM Oude akkoord wordt gewist
1680 LINE(AK-11,10)-(AK-11+16,54),4,BF
1690 RETURN
```



9.9 Een PLAY-editor

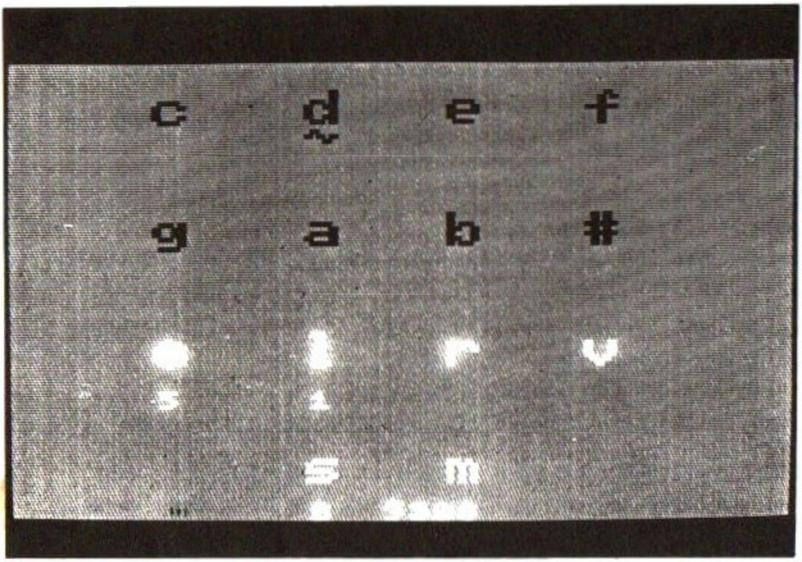
De geluids-chip van de MSX stelt u niet alleen in de gelegenheid losse tonen te produceren, maar ook om ze tegelijkertijd over drie kanalen te laten klinken. U kunt daarbij kiezen uit twee mogelijkheden: directe aansturing van de chip-registers of gebruik maken van de MSX-BASIC-instructie PLAY. Het is echter lang niet altijd even makkelijk met PLAY gecompliceerdere klanken ten gehore te brengen. Meestal zit u eerst een tijdje in het handboek te neuzen om te kijken hoe de parameters ook al weer moesten worden gebruikt. Het programma hieronder vergemakkelijkt deze werkwijze aanzienlijk. Door bijvoorbeeld een O (octaaf) of en V (volume) in te voeren kunt u met de CURSOR LINKS- en de CURSOR RECHTS-toets uw keuze maken. Het programma waakt er op die manier voor dat er geen verkeerde getallen worden ingetoetst. Om de synthesizer in te stellen (verandering van de omhullende met PLAY-parameter S) dient u die eerst met CTRL-S te activeren en na afloop met dezelfde toetsencombinatie weer uit te schakelen. Door op ENTER te drukken krijgt u de geprogrammeerde klanken te horen. De

invoer opslaan in DATA-regels heeft het voordeel dat ze met MERGE zonder problemen aan een ander BASIC-programma kunnen worden toegevoegd. Om deze opslag-optie te activeren dient u gelijk aan het begin een programmanaam in te voeren en na een toon op ESC te drukken. Het programma en de opslag van gegevens worden afgesloten met CTRL-STOP.

```
10 REM Programma 9
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM PLAY-editor; tonen kunnen worden opgeslagen in
 DATA-regels t.b.v. MERGE-mogelijkheid
50 REM Afbreekbeveiliging: DATA-files moeten worden afgesloten
60 ON STOP GOSUB 1100
70 STOP ON
80 SCREEN 1,1
90 REM PLAY-string opslaan (j/n)
100 INPUT"PLAY in DATA (j/n) ";F$
110 IF F$="j" OR F$="J" THEN INPUT "naam ";F$:TT=990:
F$="cas:"+F$:OPEN F$ FOR OUTPUT AS #1 ELSE 120
120 CLS
130 KEY OFF
140 COLOR 15,4,1
150 REM 8*8-sprites, wier uiterlijk overeenkomt met standaard
 karakterset
160 VDP(6)=VDP(4)
170 REM Beginwaarden voor PLAY-parameters
180 O=4
190 L=4
200 R=64
210 V=8
220 S=13
230 M=200
240 REM Omhullende golfvorm aan: SS=1; omhullende uit: SS=0
250 SS=0
260 REM Namen van de noten op het scherm zetten
270 PUT SPRITE 0,(46,10),1,99
280 PUT SPRITE 1,(94,10),1,100
290 PUT SPRITE 2,(140,10),1,101
300 PUT SPRITE 3,(186,10),1,102
310 PUT SPRITE 4,(46,60),1,103
320 PUT SPRITE 5,(94,60),1,97
330 PUT SPRITE 6,(140,60),1,98
340 PUT SPRITE 7,(186,60),1,35
350 REM o(ctaaf), l(engte), r(ust) en v(olume)
360 PUT SPRITE 8,(46,110),15,111
370 PUT SPRITE 9,(94,110),15,108
380 PUT SPRITE 10,(140,110),15,114
```

```
390 PUT SPRITE 11,(186,110),15,118
400 REM s (omhullende) en m (fijnafstemming omhullende)
410 PUT SPRITE 12,(94,160),10,115
420 PUT SPRITE 13,(140,160),10,109
430 REM Invoercontrole
440 A$=INKEY$
450 IF A$="" THEN GOTO 440
460 REM Wat te doen als er een noot wordt gekozen?
470 IF A$="c" THEN PUT SPRITE 14,(46,26),1,126:N$="c":GOTO 440
480 IF A$="d" THEN PUT SPRITE 14,(94,26),1,126:N$="d":GOTO 440
490 IF A$="e" THEN PUT SPRITE 14,(140,26),1,126:N$="e":GOTO 440
500 IF A$="f" THEN PUT SPRITE 14,(186,26),1,126:N$="f":GOTO 440
510 IF A$="g" THEN PUT SPRITE 14,(46,76),1,126:N$="g":GOTO 440
520 IF A$="a" THEN PUT SPRITE 14,(94,76),1,126:N$="a":GOTO 440
530 IF A$="b" THEN PUT SPRITE 14,(140,76),1,126:N$="b":GOTO 440
540 IF A$="#" THEN PUT SPRITE 14,(186,76),1,126:N$=N$+"#":GOTO 440
550 REM Wat te doen als de o (octaaf) wordt gekozen?
560 IF A$="o" THEN PUT SPRITE 14,(46,126),15,126 ELSE GOTO 630
570 IF O=8 THEN O=0
580 O=O+1
590 LOCATE 4,17
600 PRINT USING "#";O;
610 GOTO 440
620 REM Wat te doen als de l (lengte) wordt gekozen?
630 IF A$="l" THEN PUT SPRITE 14,(94,126),15,126 ELSE GOTO 700
640 IF L=64 THEN L=0
650 L=L+1
660 LOCATE 9,17
670 PRINT USING "##";L;
680 GOTO 440
690 REM Wat te doen als de r (rust) wordt gekozen?
700 IF A$="r" THEN PUT SPRITE 14,(140,126),15,126 ELSE GOTO 770
710 IF R=64 THEN R=0
720 R=R+1
730 LOCATE 15,17
740 PRINT USING "##";R;
750 GOTO 440
760 REM Wat te doen als de v (volume) wordt gekozen?
770 IF A$="v" THEN PUT SPRITE 14,(186,126),15,126 ELSE GOTO 840
780 IF V=15 THEN V=-1
790 V=V+1
800 LOCATE 21,17
810 PRINT USING "##";V;
820 GOTO 440
830 REM Wat te doen als de s (omhullende) wordt gekozen?
840 IF A$="s" THEN PUT SPRITE 14,(94,176),10,126 ELSE GOTO 910
850 IF S=14 THEN S=0
860 S=S+1
870 LOCATE 9,23
```

```
880 PRINT USING "##";S;
890 GOTO 440
900 REM Wat te doen als de m (afstemming omhullende) wordt
 gekozen?
910 IF A$="m" THEN PUT SPRITE 14,(140,176),10,126 ELSE GOTO 960
920 IF M=65500! THEN M=0
930 M=M+100
940 LOCATE 12,23
950 PRINT USING "#####";M;
960 IF A$="M" THEN PUT SPRITE 14,(140,176),10,126 ELSE GOTO 1030
970 IF M=100 THEN M=65500!
980 M=M-100
990 LOCATE 12,23
1000 PRINT USING "#####";M;
1010 GOTO 440
1020 REM Wat te doen als CTRL-S (omhullende aan/uit) wordt
 ingetoetst?
1030 IF ASC(A$)=19 THEN IF SS=0 THEN SS=1 ELSE SS=0:GOTO 440
1040 REM Wat te doen als er op ENTER (laat die toon maar eens
 horen) wordt gedrukt?
1050 IF ASC(A$)=13 THEN IF SS=1 THEN PLAY
 "o=o;l=l;r=r;v=v;s=s;m=m;xn$;" ELSE PLAY "o=o;l=l;r=r;v=v;xn$;"
1060 REM Wat te doen als ESC (geluidsgegevens opslaan) wordt
 ingedrukt?
1070 IF ASC(A$)=27 AND TT>0 THEN TT=TT+10:
A1$=STR$(TT)+" DATA "+"o"+STR$(O)+"l"+STR$(L)+"r"+STR$(R):
IF SS=0 THEN A1$=A1$+"v"+STR$(V)+N$:PRINT #1,A1$ ELSE
 A1$=A1$+"s"+STR$(S)+"m"+STR$(M)+N$:PRINT #1,A1$
1080 GOTO 440
1090 REM Subroutine voor einde programma met CTRL-STOP
1100 CLOSE
1110 SCREEN 0
1120 END
```


c d e f
g a b #

9.10 Een sound-editor

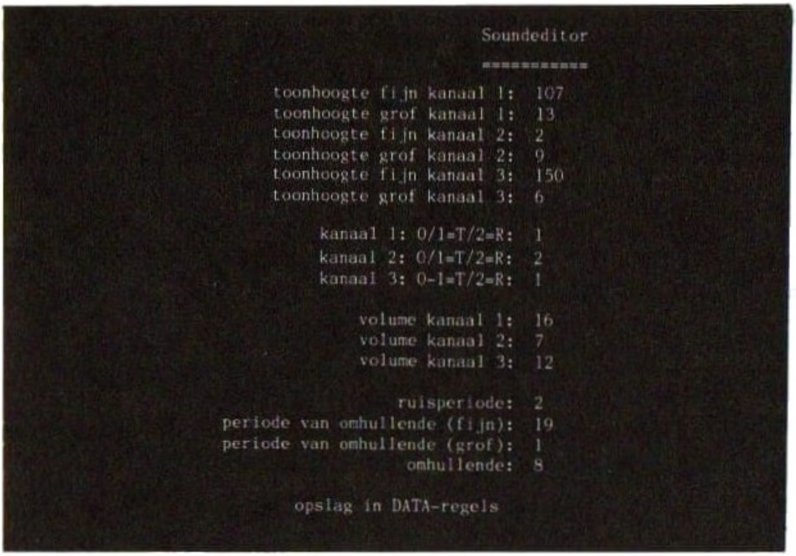
De geluids-chip van de MSX stelt u niet alleen in de gelegenheid losse tonen te produceren, maar ook om ze tegelijkertijd over drie kanalen te laten klinken. U kunt daarbij kiezen uit twee mogelijkheden: directe aansturing van de chip-registers of gebruik maken van de MSX-BASIC-instructie PLAY. Met de instructie SOUND een leuk melodietje ten gehore brengen is helemaal nog niet zo eenvoudig. Een muziekkanaal aansturen is namelijk niet voldoende. Alle voorwaarden zijn pas vervuld wanneer het volume- en combinatieregister van de juiste getallen zijn voorzien. Om vervormd geluid te creëren (met de verschillende omhullende golfvormen) moet het volume van tenminste 1 kanaal op 16 staan. Het volgende programma stelt u in staat met de cursortoetsen de registers van de geluids-chip aan te sturen (BOVEN/BENEDEN) en de waarden te veranderen (LINKS/RECHTS). Door op de spatiebalk te drukken weerklinkt de toon. De invoer opslaan in DATA-regels heeft het voordeel dat ze met MERGE zonder problemen aan een ander BASIC-programma kunnen worden toegevoegd. Om deze opslag-optie te activeren dient u gelijk aan het begin een programmanaam in te voeren en na een toon op ESC te drukken. Het programma en de opslag van gegevens worden afgesloten met CTRL-STOP.

```
10 REM Programma 10
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM Sound-editor; tonen kunnen worden opgeslagen in
 DATA-regels t.b.v. MERGE-mogelijkheid
50 REM Om een melodie op te slaan moet u aan het begin van dit
 programma een naam invoeren
60 REM Door op ESC te drukken wordt de toonopslag geëffectueerd
70 REM Afbreekbeveiliging: DATA-files moeten worden afgesloten
80 SCREEN 0:WIDTH 35
90 CLEAR 2000:DIM A$(21),A(21)
100 ON STOP GOSUB 1280
110 STOP ON
120 KEY OFF
130 COLOR 15,1,1
140 REM SOUND-parameters opslaan (j/n)
150 INPUT"SOUND in DATA (j/n) ";F$
160 IF F$="j" OR F$="J" THEN INPUT "naam ";F$:
TT=990:F$="cas:"+F$:OPEN F$ FOR OUTPUT AS #1 ELSE 170
170 CLS
180 REM
190 FOR N=0 TO 13
200 SOUND N,0
210 NEXT N
220 SOUND 7,63
230 REM Opbouw van beeldscherm
```

```
240 PRINT TAB(12) "Sound-editor"
250 PRINT TAB(12) "==========="
260 PRINT
270 PRINT TAB(1);:A$(3)="toonhoogte fijn kanaal 1:":PRINT A$(3)
280 PRINT TAB(1);:A$(4)="toonhoogte grof kanaal 1:":PRINT A$(4)
290 PRINT TAB(1);:A$(5)="toonhoogte fijn kanaal 2:":PRINT A$(5)
300 PRINT TAB(1);:A$(6)="toonhoogte grof kanaal 2:":PRINT A$(6)
310 PRINT TAB(1);:A$(7)="toonhoogte fijn kanaal 3:":PRINT A$(7)
320 PRINT TAB(1);:A$(8)="toonhoogte grof kanaal 3:":PRINT A$(8)
330 PRINT
340 PRINT TAB(6);:A$(10)="kanaal 1: 0/1=T/2=R:":PRINT A$(10)
350 PRINT TAB(6);:A$(11)="kanaal 2: 0/1=T/2=R:":PRINT A$(11)
360 PRINT TAB(6);:A$(12)="kanaal 3: 0/1=T/2=R:":PRINT A$(12)
370 PRINT
380 PRINT TAB(10);:A$(14)="volume kanaal 1:":PRINT A$(14)
390 PRINT TAB(10);:A$(15)="volume kanaal 2:":PRINT A$(15)
400 PRINT TAB(10);:A$(16)="volume kanaal 3:":PRINT A$(16)
410 PRINT
420 PRINT TAB(14);:A$(18)="ruisperiode:":PRINT A$(18)
430 PRINT TAB(4);:A$(19)="periode van omhullende (fijn):":
PRINT A$(19)
440 PRINT TAB(4);:A$(20)="periode van omhullende (grof):":
PRINT A$(20)
450 PRINT TAB(16);:A$(21)="omhullende:":PRINT A$(21)
460 IF TT<>0 THEN PRINT:PRINT "opslag in DATA-regels";
470 REM Waarden toekennen aan SOUND-parameters op scherm
480 FOR N=3 TO 8
490 LOCATE 26,N
500 PRINT A
510 NEXT N
520 FOR N=10 TO 12
530 LOCATE 26,N
540 PRINT A
550 NEXT N
560 FOR N=14 TO 16
570 LOCATE 26,N
580 PRINT A
590 NEXT N
600 FOR N=18 TO 21
610 LOCATE 26,N
620 PRINT A
630 NEXT N
640 REM Regelcursor naar juiste regel in kolom 0
650 LOCATE 0,3:PRINT CHR$(200);
660 REM Wachten op invoer
670 REM Relevante toetsen: cursorbesturing; + en - (SOUND-
 parameters); spatiebalk (geluid); ESC (opslag geluidsgegevens)
680 A$=INKEY$:IF A$="" THEN GOTO 680
690 REM Wat te doen als ESC is ingedrukt?
```

```
700 IF ASC(A$)=27 AND TT>0 THEN TT=TT+10:A1$=STR$(TT)+" DATA "+
STR$(A(3))+","+STR$(A(4))+","+STR$(A(5))+","+STR$(A(6))+","+
STR$(A(7))+","+STR$(A(8))+","+STR$(A(18))+","+STR$(A)
 ELSE GOTO 730
710 A1$=A1$+","+STR$(A(14))+","+STR$(A(15))+","+STR$(A(16))+","+
STR$(A(19))+","+STR$(A(20))+","+STR$(A(21))
720 PRINT #1,A1$
730 IF A$=" " THEN SOUND 13,A(21):GOTO 680
740 IF ASC(A$)=30 OR ASC(A$)=31 THEN GOTO 750 ELSE IF ASC(A$)=28
 OR ASC(A$)=29 THEN GOTO 830 ELSE GOTO 680
750 IF ASC(A$)=30 AND CSRLIN=3 THEN GOTO 680
760 IF ASC(A$)=31 AND CSRLIN=21 THEN GOTO 680
770 IF ASC(A$)=31 THEN IF CSRLIN=8 OR CSRLIN=12 OR CSRLIN=16
 THEN A=2 ELSE A=1 ELSE GOTO 800
780 LOCATE 0,CSRLIN:PRINT " ";:LOCATE 0,CSRLIN+A:
PRINT CHR$(200);:LOCATE 25,CSRLIN
790 GOTO 680
800 IF ASC(A$)=30 THEN IF CSRLIN=18 OR CSRLIN=14 OR CSRLIN=10
 THEN A=-2 ELSE A=-1 ELSE GOTO 680
810 LOCATE 0,CSRLIN:PRINT " ";:LOCATE 0,CSRLIN+A:
PRINT CHR$(200);:LOCATE 25,CSRLIN
820 GOTO 680
830 IF ASC(A$)=28 THEN Z=1 ELSE Z=-1
840 LOCATE 26,CSRLIN:PRINT "   ";:LOCATE 26,CSRLIN
850 REM Invoer SOUND-parameters (+/-); tevens controle of
 grenswaarden al zijn bereikt en controle van indicatie op scherm
860 REM Toonhoogte kanaal 1 (fijn)
870 IF CSRLIN=3 THEN IF (A(3)+Z<0 OR A(3)+Z>255) THEN PRINT A(3);:
GOTO 680 ELSE A(3)=A(3)+Z:PRINT A(3);:SOUND 0,A(3):GOTO 680
880 REM Toonhoogte kanaal 1 (grof)
890 IF CSRLIN=4 THEN IF (A(4)+Z<0 OR A(4)+Z>15) THEN PRINT A(4);:
GOTO 680 ELSE A(4)=A(4)+Z:PRINT A(4);:SOUND 1,A(4):GOTO 680
900 REM Toonhoogte kanaal 2 (fijn)
910 IF CSRLIN=5 THEN IF (A(5)+Z<0 OR A(5)+Z>255) THEN PRINT A(5);:
GOTO 680 ELSE A(5)=A(5)+Z:PRINT A(5);:SOUND 2,A(5):GOTO 680
920 REM Toonhoogte kanaal 2 (grof)
930 IF CSRLIN=6 THEN IF (A(6)+Z<0 OR A(6)+Z>15) THEN PRINT A(6);:
GOTO 680 ELSE A(6)=A(6)+Z:PRINT A(6);:SOUND 3,A(6):GOTO 680
940 REM Toonhoogte kanaal 3 (fijn)
950 IF CSRLIN=7 THEN IF (A(7)+Z<0 OR A(7)+Z>255) THEN PRINT A(7);:
GOTO 680 ELSE A(7)=A(7)+Z:PRINT A(7);:SOUND 4,A(7):GOTO 680
960 REM Toonhoogte kanaal 3 (grof)
970 IF CSRLIN=8 THEN IF (A(8)+Z<0 OR A(8)+Z>15) THEN PRINT A(8);:
GOTO 680 ELSE A(8)=A(8)+Z:PRINT A(8);:SOUND 5,A(8):GOTO 680
980 REM Volume kanaal 1
990 IF CSRLIN=14 THEN IF (A(14)+Z<0 OR A(14)+Z>16) THEN PRINT
 A(14);:GOTO 680 ELSE A(14)=A(14)+Z:PRINT A(14);:SOUND 8,A(14):
GOTO 680
1000 REM Volume kanaal 2
```

```
1010 IF CSRLIN=15 THEN IF (A(15)+Z<0 OR A(15)+Z>16) THEN PRINT
 A(15);:GOTO 680 ELSE A(15)=A(15)+Z:PRINT A(15);:SOUND 9,A(15):
GOTO 680
1020 REM Volume kanaal 3
1030 IF CSRLIN=16 THEN IF (A(16)+Z<0 OR A(16)+Z>16) THEN PRINT
 A(16);:GOTO 680 ELSE A(16)=A(16)+Z:PRINT A(16);:SOUND 10,A(16):
GOTO 680
1040 REM Ruisperiode
1050 IF CSRLIN=18 THEN IF (A(18)+Z<0 OR A(18)+Z>31) THEN PRINT
 A(18);:GOTO 680 ELSE A(18)=A(18)+Z:PRINT A(18);:SOUND 6,A(18):
GOTO 680
1060 REM Periode van omhullende (fijn)
1070 IF CSRLIN=19 THEN IF (A(19)+Z<0 OR A(19)+Z>255) THEN PRINT
 A(19);:GOTO 680 ELSE A(19)=A(19)+Z:PRINT A(19);:SOUND 11,A(19):
GOTO 680
1080 REM Periode van omhullende (grof)
1090 IF CSRLIN=20 THEN IF (A(20)+Z<0 OR A(20)+Z>255) THEN PRINT
 A(20);:GOTO 680 ELSE A(20)=A(20)+Z:PRINT A(20);:SOUND 12,A(20):
GOTO 680
1100 REM Omhullende
1110 IF CSRLIN=21 THEN IF (A(21)+Z<0 OR A(21)+Z>15) THEN PRINT
 A(21);:GOTO 680 ELSE A(21)=A(21)+Z:PRINT A(21);:SOUND 13,A(21):
GOTO 680
1120 REM Kanaal 1: uit, toon, ruis, beide
1130 IF CSRLIN=10 THEN IF (A(10)+Z<0 OR A(10)+Z>3) THEN PRINT
 A(10);:GOTO 680 ELSE A(10)=A(10)+Z:PRINT A(10);
1140 REM Kanaal 2: uit, toon, ruis, beide
1150 IF CSRLIN=11 THEN IF (A(11)+Z<0 OR A(11)+Z>3) THEN PRINT
 A(11);:GOTO 680 ELSE A(11)=A(11)+Z:PRINT A(11);
1160 REM Kanaal 3: uit, toon, ruis, beide
1170 IF CSRLIN=12 THEN IF (A(12)+Z<0 OR A(12)+Z>3) THEN PRINT
 A(12);:GOTO 680 ELSE A(12)=A(12)+Z:PRINT A(12);
1180 IF CSRLIN<10 OR CSRLIN>12 THEN GOTO 1260
1190 REM Precieze berekening van SOUND-parameter 7
1200 A=0
1210 IF A(10)=1 THEN A=A+8 ELSE IF A(10)=2 THEN A=A+1 ELSE IF
 A(10)=0 THEN A=A+1+8
1220 IF A(11)=1 THEN A=A+16 ELSE IF A(11)=2 THEN A=A+2 ELSE IF
 A(11)=0 THEN A=A+16+2
1230 IF A(12)=1 THEN A=A+32 ELSE IF A(12)=2 THEN A=A+4 ELSE IF
 A(12)=0 THEN A=A+32+4
1240 A=A+128
1250 SOUND 7,A
1260 GOTO 680
1270 WIDTH 40
1280 REM Subroutine voor einde programma met CTRL-STOP
1290 CLOSE
1300 SCREEN 0
1310 END
```



9.11 Een grafiek-editor

Om de grafiekinstructies uit het MSX-BASIC zinvol te kunnen gebruiken dient u normaal gesproken op millimeterpapier de posities uit te rekenen die moeten worden ingevoerd. Het onderstaande programma maakt die handelwijze overbodig. U beweegt gewoon een kruis over het scherm zonder zich ook maar iets aan te trekken van coördinaten en kiest daarmee een bepaald punt. Het programma heeft de functietoetsen gedefinieerd als BASIC-instructies, de grafische mogelijkheden van de MSX kunnen dus zonder problemen en heel gemakkelijk worden uitgebuit.

F1	cirkel trekken	F2	lijn trekken
F3	tekenkleur wijzigen	F4	tekst op grafiek
F5	tekenen ja/nee	F6	plaatje opslaan
F7	plaatje laden	F8	kaderkleur wijzigen
F9	vlakken inkleuren	F10	delen kopiëren

Laden en opslaan in een fractie van een seconde kunt u op deze manier wel vergeten. Wen maar alvast aan het idee dat het eerder een kwestie van vele minuten is.

Bij de kopieer-optie komt nog wat meer kijken dan alleen F10. Met het kruis met de witte kern geeft u de linker benedenhoek aan van het veld dat gekopieerd moet worden, daarna drukt u op F10 en

geeft met het kruis de rechter bovenhoek aan. De witte kern blijft op de coördinaten van de linker benedenhoek staan. Druk daar aangekomen op ENTER. Dat heeft tot gevolg dat de computer de rechter bovenhoek onthoudt. Druk vervolgens op een cursortoets en het kruis springt weer naar de witte kern terug. Van daaruit kunt u met de cursorbesturing de plek op het scherm opzoeken waar u de kopie wilt hebben staan. Door op ENTER te drukken wordt de kopie daadwerkelijk gemaakt.

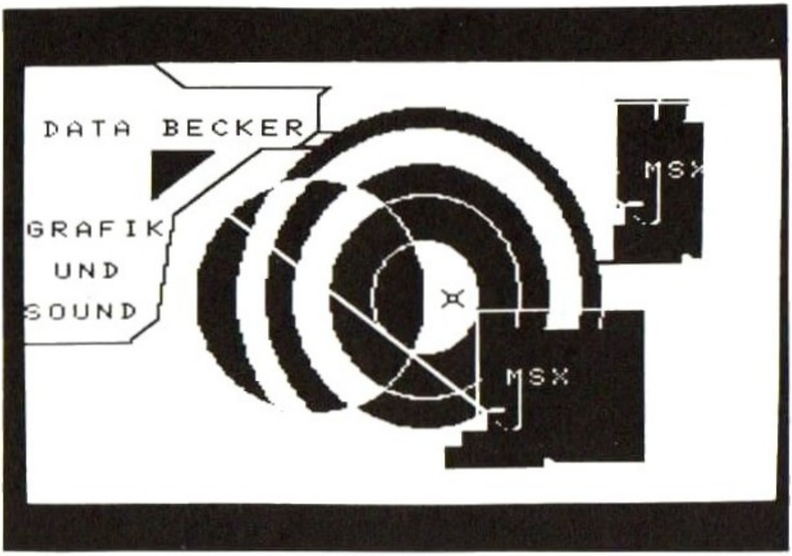
```
10 REM Programma 11
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM Grafiek-editor; besturing via functietoetsen
50 REM Plaatjes kunnen op band worden gezet; gedeelten van de
 grafische voorstellingen kunnen worden gekopieerd
60 REM Afbreekbeveiliging met ON STOP GOSUB om te voorkomen dat
 er een onvoorziene kleurencombinatie op het scherm verschijnt
70 ON STOP GOSUB 1990
80 STOP ON
90 REM F1: cirkel tekenen als 2e cursor is ingesteld; F2: lijn
 tekenen als 2e cursor is ingesteld; F3: tekenkleur veranderen
 (getal van 2 cijfers invoeren)
100 REM F4: tekst op cursorpositie neerzetten; <ENTER> betekent
 einde
110 REM F5: wel/niet tekenen, komt overeen met beweging van
 cursor; F6: opslaan met naam en titel (duur: ca. 10 minuten);
 F7: plaatjes laden (naam invoeren)
120 REM F8: kaderkleur veranderen (2 cijfers invoeren, b.v. 08);
 geen <ENTER> gebruiken
130 REM F9: vlakken inkleuren met de actuele kleur; PAS OP dat de
 cursor zich niet buiten het figuurtje bevindt anders bent u dit
 (en evt. ook andere) kwijt: buiten de figuurtjes wordt er
 namelijk doorgekleurd tot aan het kader
140 REM F10: delen van het scherm kopiëren: linksonder
 (afsluiten met F10); rechtsboven (afsluiten met ENTER); met
 cursortoetsen nieuwe plek kiezen (= linker benedenhoek)
 (afsluiten met ENTER)
150 ON KEY GOSUB 550,680,810,850,930,960,1190,1410,1440,1470
160 REM Voorbereidingen treffen
170 COLOR 15,4,5
180 SCREEN 2,2
190 FOR N=1 TO 10
200 KEY(N) ON
210 NEXT N
220 REM Kruis-sprite inlezen, die als grafiekcursor fungeert
230 FOR N=1 TO 8
240 READ A$
250 A$="&b"+A$
260 B$=B$+CHR$(VAL(A$))
```

```
270 NEXT N
280 SPRITE$(1)=B$
290 REM Middelpunt van grafiekcursor inlezen, zodat die op elke
 achtergrond te zien is
300 FOR N=1 TO 8
310 READ A$
320 A$="&B"+A$
330 C$=C$+CHR$(VAL(A$))
340 NEXT N
350 SPRITE$(2)=C$
360 REM Zwarte grafiekcursor op midden van scherm zetten
370 A=128
380 B=96
390 F=1
400 COLOR F
410 PUT SPRITE 1,(A,B),1
420 PUT SPRITE 2,(A,B),15
430 REM Moeten er punten worden gezet?
440 IF ME=1 THEN PSET(A+4,B+4),F
450 REM Richtingscontrole m.b.t. cursortoetsen
460 A$=INKEY$
470 IF A$="" THEN GOTO 460
480 IF ASC(A$)=28 THEN IF A<>250 THEN A=A+1
490 IF ASC(A$)=29 THEN IF A<>-4 THEN A=A-1
500 IF ASC(A$)=30 THEN IF B<>-4 THEN B=B-1
510 IF ASC(A$)=31 THEN IF B<>186 THEN B=B+1
520 PUT SPRITE 1,(A,B),1
530 PUT SPRITE 2,(A,B),15
540 GOTO 410
550 REM Cirkel trekken
560 C=A
570 D=B
580 REM Richting inlezen (cursortoetsen)
590 A$=INKEY$
600 IF A$="" THEN GOTO 590
610 IF ASC(A$)=28 THEN IF C<>250 THEN C=C+1
620 IF ASC(A$)=29 THEN IF C<>-4 THEN C=C-1
630 IF ASC(A$)=30 THEN IF D<>-4 THEN D=D-1
640 IF ASC(A$)=31 THEN IF D<>186 THEN D=D+1
650 IF A$=CHR$(13) THEN A1=ABS(A-C):A2=ABS(B-D):IF A1>=A2 THEN
 CIRCLE (A+4,B+4),A1+13,F,,,4/3:RETURN ELSE
 CIRCLE (A+4,B+4),A2,F,,,4/3:RETURN
660 PUT SPRITE 1,(C,D),1
670 GOTO 590
680 REM Lijn trekken
690 C=A
700 D=B
710 REM Richting inlezen (cursortoetsen)
720 A$=INKEY$
```

```
730 IF A$="" THEN GOTO 720
740 IF ASC(A$)=28 THEN IF C<>250 THEN C=C+1
750 IF ASC(A$)=29 THEN IF C<>-4 THEN C=C-1
760 IF ASC(A$)=30 THEN IF D<>-4 THEN D=D-1
770 IF ASC(A$)=31 THEN IF D<>186 THEN D=D+1
780 IF A$=CHR$(13) THEN LINE (A+4,B+4)-(C+4,D+4),F:RETURN
790 PUT SPRITE 1,(C,D),1
800 GOTO 720
810 REM Kleur veranderen
820 F$=INPUT$(2)
830 F=VAL(F$)
840 IF F<16 THEN COLOR F:RETURN ELSE RETURN
850 REM Tekst invoeren
860 OPEN "grp:"FOR OUTPUT AS #1
870 DRAW "bm =a;,=b;"
880 A$=INKEY$
890 IF A$="" THEN GOTO 880
900 IF A$=CHR$(13) THEN CLOSE:RETURN
910 PRINT #1,A$;
920 GOTO 880
930 REM Wel/niet tekenen
940 IF ME=1 THEN ME=0 ELSE ME=1
950 RETURN
960 REM Opslaan
970 REM Door gegevens weg te schrijven kan het niet tot
 ongewenste kleurvermenging komen
980 LINE(100,15)-(160,0),1,BF
990 OPEN "grp:" FOR OUTPUT AS #1
1000 NN$=""
1010 COLOR 15
1020 DRAW "bm 108,4"
1030 REM Invoer van programmanaam/titel (maximaal 6 tekens)
1040 FOR N=1 TO 6
1050 NN$=NN$+N$
1060 N$=INPUT$(1)
1070 IF N$=CHR$(13) THEN GOTO 1080 ELSE PRINT #1,N$;:NEXT N
1080 A$=INKEY$
1090 REM Beveiliging
1100 IF A$="" THEN GOTO 1080 ELSE IF A$<>CHR$(13) THEN RETURN
1110 CLOSE #1
1120 OPEN NN$ FOR OUTPUT AS #1
1130 REM Het 16K video-RAM opslaan
1140 FOR N=0 TO 16383
1150 PRINT #1,VPEEK(N)
1160 NEXT N
1170 CLOSE
1180 RETURN
1190 REM Laden
1200 REM Door gegevens weg te schrijven kan het niet tot
```

```
 ongewenste kleurvermenging komen
1210 LINE(100,15)-(160,0),1,BF
1220 OPEN "grp:" FOR OUTPUT AS #1
1230 NN$=""
1240 COLOR 15
1250 DRAW "bm 108,4"
1260 REM Invoer van programmanaam/titel (maximaal 6 tekens)
1270 FOR N=1 TO 6
1280 NN$=NN$+N$
1290 N$=INPUT$(1)
1300 IF N$=CHR$(13) THEN GOTO 1080 ELSE PRINT #1,N$;:NEXT N
1310 REM Beveiliging
1320 A$=INKEY$
1330 IF A$="" THEN GOTO 1320 ELSE IF A$<>CHR$(13) THEN RETURN
1340 CLOSE #1
1350 REM Het 16K video-RAM laden
1360 OPEN NN$ FOR INPUT AS #1
1370 FOR N=0 TO 16383
1380 INPUT #1,D
1390 VPOKE N,D
1400 NEXT N
1410 REM Border veranderen
1420 A$=INPUT$(2)
1430 IF VAL(A$)<16 THEN COLOR ,,VAL(A$):RETURN ELSE RETURN
1440 REM Inkleuren
1450 PAINT (A,B),F
1460 RETURN
1470 REM Verdubbelen (kopiëren)
1480 C1=A
1490 D1=B
1500 REM Rechter bovenhoek
1510 REM Richting inlezen (cursortoetsen)
1520 A$=INKEY$
1530 IF A$="" THEN GOTO 1520
1540 IF ASC(A$)=28 THEN IF C1<>250 THEN C1=C1+1
1550 IF ASC(A$)=29 THEN IF C1<>-4 THEN C1=C1-1
1560 IF ASC(A$)=30 THEN IF D1<>-4 THEN D1=D1-1
1570 IF ASC(A$)=31 THEN IF D1<>186 THEN D1=D1+1
1580 IF A$=CHR$(13) THEN GOTO 1620
1590 PUT SPRITE 1,(C1,D1),1
1600 GOTO 1520
1610 REM Ten opzichte van de oorsprong moet de cursor rechtsboven
 worden neergezet
1620 IF A>C1 OR B<D1 THEN BEEP:RETURN ELSE C2=A:D2=B
1630 REM Linker benedenhoek
1640 REM Richting inlezen (cursortoetsen)
1650 A$=INKEY$
1660 IF A$="" THEN GOTO 1650
1670 IF ASC(A$)=28 THEN IF C2<>250 THEN C2=C2+1
```

```
1680 IF ASC(A$)=29 THEN IF C2<>-4 THEN C2=C2-1
1690 IF ASC(A$)=30 THEN IF D2<>-4 THEN D2=D2-1
1700 IF ASC(A$)=31 THEN IF D2<>186 THEN D2=D2+1
1710 IF A$=CHR$(13) THEN GOTO 1750
1720 PUT SPRITE 1,(C2,D2),1
1730 GOTO 1650
1740 REM Kopieerproces
1750 FOR N=A TO C1
1760 FOR M=B TO D1 STEP-1
1770 PSET(C2+(N-A),D2+(M-B)),POINT(N,M)
1780 NEXT M,N
1790 RETURN
1800 REM Sprite 1
1810 DATA 10000001
1820 DATA 01000010
1830 DATA 00111100
1840 DATA 00100100
1850 DATA 00100100
1860 DATA 00111100
1870 DATA 01000010
1880 DATA 10000001
1890 REM Sprite 2
1900 DATA 00000000
1910 DATA 00000000
1920 DATA 00000000
1930 DATA 00011000
1940 DATA 00011000
1950 DATA 00000000
1960 DATA 00000000
1970 DATA 00000000
1980 REM Bij afbreking van het programma krijgen we weer een
 leesbaar kleurcontrast
1990 COLOR 15,1
```



9.12 Karakterset vergroot printen

Helemaal aan het eind van dit boek staat de karakterset van de MSX: gewone cijfers en letters maar ook computertekens. Het thema 'wijzigingen aanbrengen in de karakterset' komt in dit boek uitvoerig aan de orde. In paragraaf 9.14 passeert zelfs een behoorlijk professionele karaktergenerator de revue. Daarom is het zinvol de printroutine enigzins uit te breiden en apart te behandelen. Na de karaktergenerator te hebben gebruikt kunt u dan het onderstaande programma laden. De karakterset wordt geen haar gekrenkt: video-RAM en programmageheugen liggen immers heel ergens anders. Het volgende printprogramma is geschreven voor een niet-MSX-printer, de Epson MX-80. Bezitters van een MSX-printer moeten daarom het Epson-teken CHR$(233) veranderen in het MSX-teken CHR$(219). We werken met een standaard papierlengte (72 regels per pagina). Denk erom dat het aantal te printen tekennummers deelbaar moet zijn door 30 (bijvoorbeeld: het interval 129 tot en met 218 beslaat 90 tekens; dat komt neer op 30 regels met 3 tekens, dus 7,5 pagina's met 12 tekens elk). Het programma functioneert alleen als u zich al in SCREEN 1 bevindt. Elke SCREEN-instructie op een later tijdstip wist het video-RAM en dus de karakterset.

```
10 REM Programma 12
20 SCREEN(1)
30 REM Een Lüers-programma
40 REM Programma om de karakterset vergroot uit te printen
50 REM Oorspronkelijk bedoeld voor een Epson; Epson-blok:
 CHR$(223), MSX-blok: CHR$(219)
60 REM Na 12 tekens (4 kolommen van 3) gaan we verder op de
 volgende bladzijde; dat moet worden gewijzigd als u niet met A4
 (72 regels) werkt
70 CLS
80 PRINT"U moet zich in "
90 PRINT"SCREEN(1) bevinden"
100 PRINT
110 PRINT"Welk teken moet als"
120 PRINT"eerste geprint"
130 INPUT"worden (1 t/m 255)";A
140 PRINT
150 PRINT"Welk teken moet als"
160 PRINT"laatste geprint"
170 PRINT"worden (";RIGHT$(STR$(A),(LEN(STR$(A))-1));
180 INPUT"1 t/m 255) ";B
190 REM Controle; o.a. of de gekozen tekens in de driekoloms
 uitdraai passen
200 IF A>B OR B>255 OR A<0 OR (B+1-A)/3<>INT((B+1-A)/3) THEN RUN
210 PRINT
220 PRINT"Dat zijn de"
230 PRINT"tekens die"
240 PRINT"geprint worden:"
250 PRINT
260 FOR N=A TO B
270 PRINT CHR$(N);
280 NEXT N
290 PRINT
300 INPUT"Okee, druk dan op <ENTER> ";A$
310 IF A$<>"" THEN RUN
320 REM Geheugenplaatsen waar in SCREEN 1 de CHR$-tekens liggen
 opgeslagen
330 FOR N=A*8 TO B*8 STEP 24
340 LPRINT
350 REM Linker teken printen
360 LPRINT"karakterteken";INT(N/8);
370 REM Middelste teken printen
380 LPRINT TAB(29)"karakterteken";INT((N+8)/8);
390 REM Rechter teken printen
400 LPRINT TAB(58)"karakterteken";INT((N+16)/8)
410 LPRINT
420 LPRINT TAB(8)"76543210";
430 LPRINT TAB(37);"76543210";
440 LPRINT TAB(66);"76543210"
```

```
450 LPRINT
460 FOR M=0 TO 7
470 REM Decimale representatie veranderen in volledige 8-bits
 binaire representatie
480 A$=BIN$(VPEEK(N+M))
490 B$=BIN$(VPEEK(N+M+8))
500 C$=BIN$(VPEEK(N+M+16))
510 A$=STRING$(8-LEN(A$),"0")+A$
520 B$=STRING$(8-LEN(B$),"0")+B$
530 C$=STRING$(8-LEN(C$),"0")+C$
540 LPRINT TAB(5)M;TAB(7);
550 FOR L=1 TO 8
560 REM Binaire reeks omwerken tot gevulde (enen) en lege
 (nullen) blokken
570 IF MID$(A$,L,1)="1" THEN LPRINT CHR$(223);ELSE LPRINT " ";
580 NEXT L
590 LPRINT TAB(34)M;TAB(36);
600 FOR L=1 TO 8
610 IF MID$(B$,L,1)="1" THEN LPRINT CHR$(223);ELSE LPRINT" ";
620 NEXT L
630 LPRINT TAB(63)M;TAB(65);
640 FOR L=1 TO 8
650 IF MID$(C$,L,1)="1" THEN LPRINT CHR$(223);ELSE LPRINT" ";
660 NEXT L
670 LPRINT
680 NEXT M
690 LPRINT
700 REM Nieuwe pagina al nodig?
710 Z=Z+1
720 IF Z=4 THEN Z=0:FOR U=1 TO 16:LPRINT:NEXT
730 NEXT N
740 RUN
```



9.13

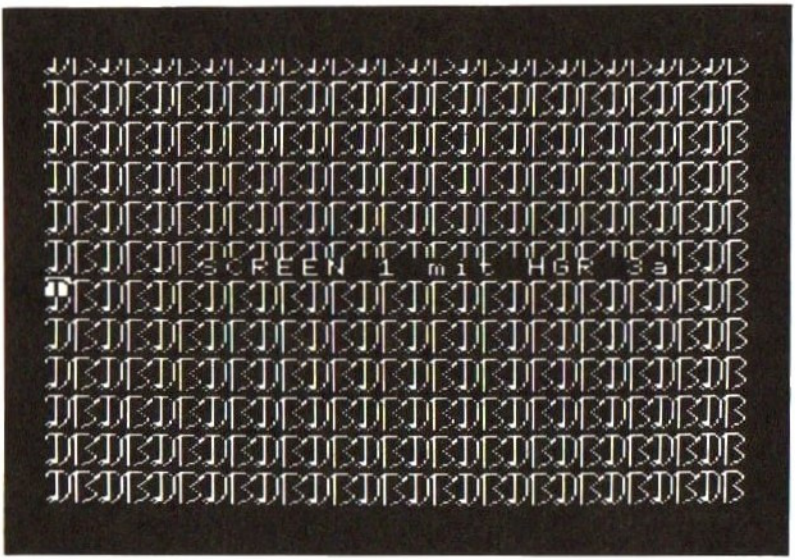
De MSX heeft twee text screens (SCREEN 0 (40 tekens per regel en SCREEN 1 (32 tekens per regel)) en twee grafiekschermen (SCREEN 2 (256*192 punten) en SCREEN 3 (64*48 punten)). Op de grafiekschermen kan ook tekst worden gezet, maar grafiek op een tekstscherm kan alleen worden gerealiseerd met sprites op SCREEN 1. Op de tekstschermen SCREEN 0 en 1 kunnen tegelijkertijd maximaal 256 verschillende tekens worden neergezet. Als de vorm van een der tekens wordt veranderd, heeft dat effect op alle tekens die op het scherm die vorm hebben. In de paragrafen 4.1 en 4.2 over de instructies CHR$ en ASC vindt u meer informatie. Het demonstratie-programma in deze paragraaf wekt de indruk dat er grafiek op het tekstscherm verschijnt. Maximaal 4 tekens worden voortdurend veranderd. Draai het programma en bestudeer de listing. Met wat fantasie kunt u zelf vast ook wel iets dergelijks maken.

```
10 REM Programma 13
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM Programma om op SCREEN 1 grafiek-achtige effecten te
 creëren
50 REM Door het beginadres van de karakterset in het VRAM te
 wijzigen kunt u dit resultaat ook in SCREEN 0 bereiken
60 REM Raadpleeg paragraaf 9.13 voor nadere informatie
70 REM Voorbereidingen treffen
80 SCREEN 1
90 KEY OFF
100 REM Karakter 250 wissen
110 REM (1) een punt draait linksom rond
120 GOSUB 980
130 REM Linksboven komt karakter 250 te staan
140 LOCATE 0,0
150 PRINT CHR$(250)
160 LOCATE 6,10
170 PRINT "SCREEN 0 met HGR 1"
180 FOR L=1 TO 10
190 REM Beweging van punt
200 GOSUB 870
210 REM Karakter 250 wissen
220 GOSUB 980
230 NEXT L
240 REM (2) een scherm vol linksom ronddraaiende punten
250 GOSUB 1300
260 REM Het scherm wordt volgezet met karakter 250
270 LOCATE 6,10
280 PRINT "SCREEN 0 met HGR 1a"
290 FOR L=1 TO 10
300 REM Beweging van de punten
310 GOSUB 870
320 REM Karakter 250 wissen
330 GOSUB 980
340 NEXT L
350 REM (3) steeds hetzelfde vormpje wordt getekend
360 CLS
370 LOCATE 0,0
380 PRINT CHR$(250)
390 LOCATE 6,10
400 PRINT "SCREEN 0 met HGR 2"
410 FOR L=1 TO 10
420 REM Opbouw van de tekening
430 GOSUB 1030
440 REM Teken wissen
450 GOSUB 980
460 NEXT L
470 REM (4) overal op het scherm wordt steeds hetzelfde vormpje
```

```
 getekend
480 GOSUB 1300
490 REM Overal op het scherm komt karakter 250 te staan
500 LOCATE 6,10
510 PRINT "SCREEN 0 met HGR 2a"
520 FOR L=1 TO 10
530 REM Opbouw van de tekening
540 GOSUB 1030
550 REM Teken wissen
560 GOSUB 980
570 NEXT L
580 REM (5) de karakters 250-253 worden steeds opnieuw op het
 scherm gezet
590 CLS
600 LOCATE 0,0
610 REM Teken wissen
620 GOSUB 1230
630 PRINT CHR$(250);CHR$(252)
640 PRINT CHR$(251);CHR$(253)
650 LOCATE 6,10
660 PRINT "SCREEN 1 met HGR 3"
670 FOR L=1 TO 10
680 REM Teken inlezen en afbeelden
690 GOSUB 1120
700 REM Teken wissen
710 GOSUB 1230
720 NEXT L
730 REM (6) de karakters 250-253 worden overal op het scherm
 steeds opnieuw getekend
740 CLS
750 REM Overal op het scherm de karakters 250-253 neerzetten
760 GOSUB 1370
770 LOCATE 6,10
780 PRINT "SCREEN 1 met HGR 3a"
790 FOR L=1 TO 10
800 REM Teken inlezen en afbeelden
810 GOSUB 1120
820 REM Teken wissen
830 GOSUB 1230
840 NEXT L
850 END
860 REM Subroutine: tekenvorm inlezen voor (1) en (2)
870 RESTORE 1280
880 FOR N=1 TO 20
890 READ A,B
900 VPOKE(250*8+A),B
910 REM Punt weer wissen
920 FOR M=1 TO 10
930 NEXT M
```

```
940 VPOKE(250*8+A),0
950 NEXT N
960 RETURN
970 REM Subroutine: teken wissen
980 FOR N=250*8+0 TO 250*8+7
990 VPOKE N,0
1000 NEXT N
1010 RETURN
1020 REM Subroutine: tekenvorm inlezen voor (3) en (4)
1030 RESTORE 1280
1040 FOR N=1 TO 20
1050 READ A,B
1060 B=VPEEK(250*8+A) OR B
1070 VPOKE(250*8+A),B
1080 FOR M=1 TO 10
1090 NEXT M
1100 NEXT N
1110 RETURN
1120 REM Subroutine: tekenvorm inlezen voor (5) en (6)
1130 RESTORE 1490
1140 FOR N=1 TO 73
1150 READ A,B,C
1160 C=VPEEK(250*8+A*8+B) OR C
1170 VPOKE(250*8+A*8+B),C
1180 FOR M=1 TO 10
1190 NEXT M
1200 NEXT N
1210 RETURN
1220 REM Subroutine: tekens wissen (karakters 250-253)
1230 FOR N=250*8 TO 250*8+32
1240 VPOKE N,0
1250 NEXT N
1260 RETURN
1270 REM DATA voor karakter 250 [(1)-(4)]
1280 DATA 0,128,1,64,2,32,3,16,4,16,5,32,6,32,7,16,
6,8,6,4,6,2,5,1,4,1,3,2,2,4,1,4,0,8,0,16,0,32,0,64
1290 REM Subroutine: scherm vullen met karakter 250
1300 FOR N=0 TO 28
1310 FOR M=0 TO 23
1320 LOCATE N,M
1330 PRINT CHR$(250);
1340 NEXT M,N
1350 RETURN
1360 REM Subroutine: scherm vullen met karakter 250-253
1370 FOR L=1 TO 12
1380 FOR N=0 TO 13
1390 PRINT CHR$(250);CHR$(252);
1400 NEXT N
1410 PRINT
```

```
1420 FOR N=0 TO 13
1430 PRINT CHR$(251);CHR$(253);
1440 NEXT N
1450 PRINT
1460 NEXT L
1470 RETURN
1480 REM DATA voor karakter 250-253
1490 DATA 0,2,16,0,3,16,0,4,16,0,5,16,0,6,16,0,7,16,1,0,16,1,1,
16,1,2,16,1,3,16,1,4,32,1,4,64,1,5,128,1,6,64,1,5,32,1,4,16,1,5,
16,1,6,8,1,5,4,1,4,2,1,3,2,1,2,1,1,1,1,1,0,1,0,7,1,0,6,1,0,5,1,
0,4,1,0,3,1
1500 DATA 0,2,2,0,1,4,0,1,8,0,1,16,0,1,32,0,1,64,0,2,128,0,3,128
1510 DATA 2,2,64,2,3,64,2,4,64,2,5,64,2,6,64,2,7,64,3,0,64,3,1,
64,3,2,64,3,3,64,3,4,64,3,5,64,3,6,64,2,2,128,2,1,64,2,1,32,2,1,
16,2,1,8,2,1,4,2,2,2,2,3,1,2,4,1,2,5,2,2,6,4,2,7,8,3,0,4,3,1,2,
3,2,1
1520 DATA 3,3,1,3,4,2,3,5,4,3,5,8,3,6,16,3,6,32,3,6,64,3,6,128
```



9.14 Een karaktergenerator

De instructies VPEEK en VPOKE kunt u dankzij het onderstaande programma in het vervolg wel achterwege laten. Het stelt u in staat op een hele makkelijke manier de karakterset te wijzigen. Bij een karaktergenerator kun je twee dingen doen: ofwel elk teken dat kan worden veranderd voorzien van een aparte hulptekst en zijn eigen getal, ofwel de karakters in grotere groepen indelen. Voor dat laatste is hier gekozen. We werken met slechts drie tekens: de spatie, de verticale streep en de punt. Die drie kunt u met het onderstaande programma niet wijzigen. De punt geeft de eenheden, tien- en honderdtallen aan, de streep scheidt die categorieën en de spatie geeft aan dat een bepaalde categorie niet voorkomt. Het getal 214 komt er op deze manier als volgt uit te zien:

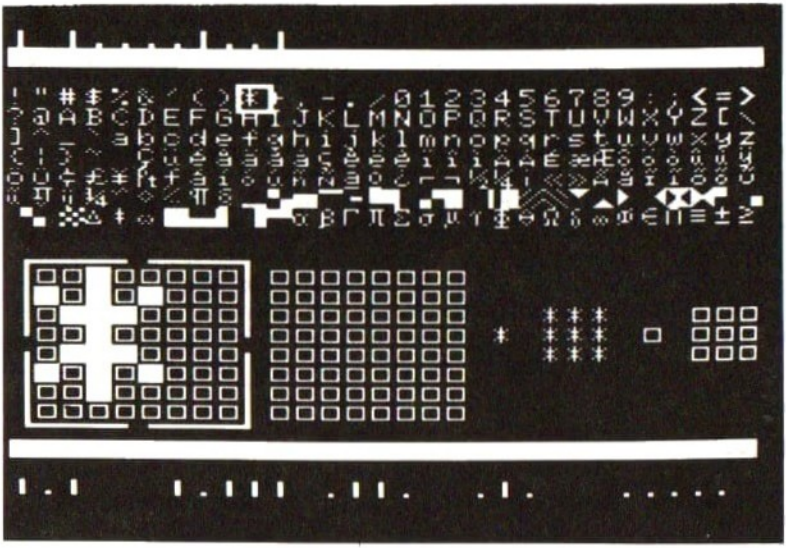
..|.|....|

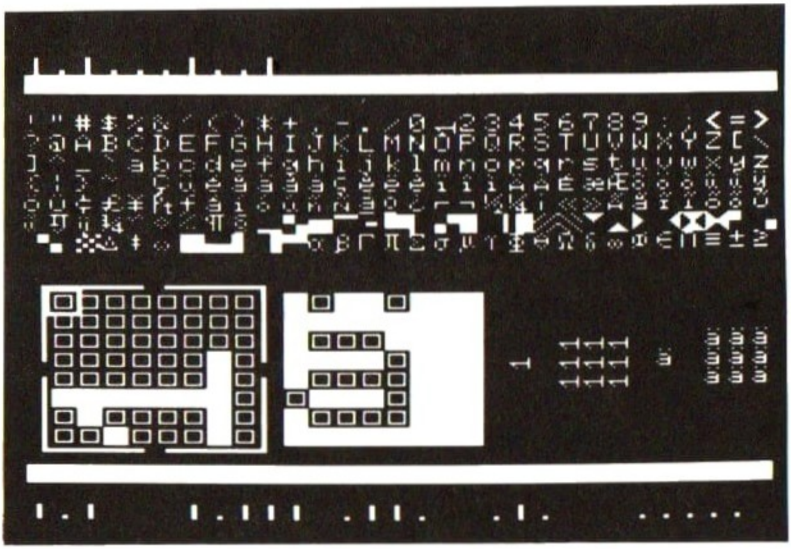
2 punten voor de honderdtallen, afgesloten door de streep
1 punt voor de tientallen, afgesloten door de streep
4 punten voor de eenheden, afgesloten door de streep

De betekenis van de functietoetsen in dit programma:

F1/F6	streep, punt, streep: karakterset laden/opslaan
F2/F7	3x streep, punt, streep: karakters 1/2
F3	punt, 2x streep, punt: spiegeling
F4	punt, streep, punt: rotatie naar rechts
F5	5x punt: hulpfaciliteit aan/uit
F8	streep, spatie, streep: wissen
F9	punt, streep, spatie, streep: invers
F10	spatie, streep, spatie, 2x streep: wisseling

Met de TAB-toets komt u in de editor, waar u met de spatiebalk punten kunt zetten en weghalen. ENTER effectueert de wijziging en met ESC kiest u de karakterset weer.





```
10 REM Programma 14
20 REM MSX grafiek en geluid
30 REM Een Lüers-programma
40 REM Karaktergenerator
50 REM Een eigen karakterset creëert u hiermee in een handomdraai
60 REM Het werkscherm begint op VRAM-adres 6144
70 VDP(2)=6
80 REM De karakterset waar u mee werkt begint op adres 0
90 VDP(4)=0
100 DEFINT A-Z
110 CLEAR 2000
120 DIM H$(13)
130 REM Om het programma af te kunnen breken moet eerst het scherm
 zijn opgeruimd
140 ON STOP GOSUB 5530
150 STOP ON
160 REM Begin van de subroutines
170 ON KEY GOSUB 3600,3710,3800,4180,4560,4590,4700,4790,5070,
5350
180 FOR N=1 TO 10
190 KEY(N) ON
200 NEXT N
210 REM Definitie van de karakters 251-254 t.b.v. functietoetsen
 en weergave van getallen
220 DATA 00000000
230 DATA 01111110
240 DATA 01000010
250 DATA 01000010
260 DATA 01000010
270 DATA 01000010
280 DATA 01111110
290 DATA 00000000
300 '
310 DATA 11111111
320 DATA 11111111
330 DATA 11111111
340 DATA 11111111
350 DATA 11111111
360 DATA 11111111
370 DATA 11111111
380 DATA 11111111
390 '
400 DATA 00000000
410 DATA 00000000
420 DATA 00000000
430 DATA 00000000
440 DATA 00000000
450 DATA 00000000
460 DATA 00011000
```

```
470 DATA 00011000
480 '
490 DATA 00000000
500 DATA 00011000
510 DATA 00011000
520 DATA 00011000
530 DATA 00011000
540 DATA 00011000
550 DATA 00011000
560 DATA 00011000
570 '
580 REM Blokkader: 4 16*16-sprites
590 DATA 1111111111111111
600 DATA 1000000000000000
610 DATA 1000000000000000
620 DATA 1000000000000000
630 DATA 1000000000000000
640 DATA 1000000000000000
650 DATA 1000000000000000
660 DATA 1000000000000000
670 DATA 1000000000000000
680 DATA 1000000000000000
690 DATA 1000000000000000
700 DATA 1000000000000000
710 DATA 1000000000000000
720 DATA 1000000000000000
730 DATA 1000000000000000
740 DATA 1000000000000000
750 '
760 DATA 1111111111111111
770 DATA 0000000000000001
780 DATA 0000000000000001
790 DATA 0000000000000001
800 DATA 0000000000000001
810 DATA 0000000000000001
820 DATA 0000000000000001
830 DATA 0000000000000001
840 DATA 0000000000000001
850 DATA 0000000000000001
860 DATA 0000000000000001
870 DATA 0000000000000001
880 DATA 0000000000000001
890 DATA 0000000000000001
900 DATA 0000000000000001
910 DATA 0000000000000001
920 '
930 DATA 1000000000000000
940 DATA 1000000000000000
950 DATA 1000000000000000
```

```
960 DATA 1000000000000000
970 DATA 1000000000000000
980 DATA 1000000000000000
990 DATA 1000000000000000
1000 DATA 1000000000000000
1010 DATA 1000000000000000
1020 DATA 1000000000000000
1030 DATA 1000000000000000
1040 DATA 1000000000000000
1050 DATA 1000000000000000
1060 DATA 1000000000000000
1070 DATA 1000000000000000
1080 DATA 1111111111111111
1090 '
1100 DATA 0000000000000001
1110 DATA 0000000000000001
1120 DATA 0000000000000001
1130 DATA 0000000000000001
1140 DATA 0000000000000001
1150 DATA 0000000000000001
1160 DATA 0000000000000001
1170 DATA 0000000000000001
1180 DATA 0000000000000001
1190 DATA 0000000000000001
1200 DATA 0000000000000001
1210 DATA 0000000000000001
1220 DATA 0000000000000001
1230 DATA 0000000000000001
1240 DATA 0000000000000001
1250 DATA 1111111111111111
1260 '
1270 REM Sprite om teken te kiezen en te veranderen
1280 DATA 1111110000000000
1290 DATA 1000010000000000
1300 DATA 1000010000000000
1310 DATA 1000010000000000
1320 DATA 1000010000000000
1330 DATA 1111110000000000
1340 DATA 0000000000000000
1350 DATA 0000000000000000
1360 DATA 0000000000000000
1370 DATA 0000000000000000
1380 DATA 0000000000000000
1390 DATA 0000000000000000
1400 DATA 0000000000000000
1410 DATA 0000000000000000
1420 DATA 0000000000000000
1430 DATA 0000000000000000
1440 '
```

```
1450 REM Codering van de 10 functietoetsen
1460 KEY 1,CHR$(254)+CHR$(253)+CHR$(254)
1470 KEY 2,CHR$(254)+CHR$(253)+CHR$(254)+CHR$(254)+CHR$(254)
1480 KEY 3,CHR$(253)+CHR$(254)+CHR$(254)+CHR$(253)
1490 KEY 4,CHR$(253)+CHR$(254)+CHR$(253)
1500 KEY 5,STRING$(5,CHR$(253))
1510 KEY 6,CHR$(254)+CHR$(253)+CHR$(253)+CHR$(254)
1520 KEY 7,CHR$(254)+CHR$(254)+CHR$(254)+CHR$(253)+CHR$(254)
1530 KEY 8,CHR$(254)+" "+CHR$(254)
1540 KEY 9,CHR$(253)+CHR$(254)+" "+CHR$(254)
1550 KEY 10," "+CHR$(254)+" "+CHR$(254)+CHR$(254)
1560 REM SCREEN 1; vergrote 16*16-sprites
1570 SCREEN 1,3
1580 COLOR 15,1
1590 WIDTH 30
1600 KEY ON
1610 PRINT"Even geduld a.u.b."
1620 REM Tekencode laden die bestaat uit de karakters 251-254
1630 FOR N=0 TO 3
1640 FOR M=0 TO 7
1650 READ A$
1660 A=VAL("&B"+A$)
1670 VPOKE (BASE(7)+((251+N)*8))+M,A
1680 NEXT M
1690 NEXT N
1700 REM Karakterset kopiëren in de VRAM-adressen 10240 e.v.
 (VDP(4)=5)
1710 FOR N%=0 TO 2048
1720 VPOKE N%+10240,VPEEK(N%)
1730 NEXT N%
1740 REM Tweede scherm als hulpscherm opslaan (VDP(2)=9)
1750 H$(0)="  Definiëring functietoetsen"
1760 H$(1)="  "+STRING$(23,219)
1770 H$(2)=STRING$(32," ")
1780 H$(3)=STRING$(32," ")
1790 H$(4)="  1 = Karakterset laden"
1800 H$(5)="  2 = Karakterblok 1"
1810 H$(6)="  3 = Spiegeling"
1820 H$(7)="  4 = Rotatie naar rechts"
1830 H$(8)="  5 = HELP aan/uit"
1840 H$(9)="  6 = Karakterset opslaan"
1850 H$(10)="  7 = Karakterblok 2"
1860 H$(11)="  8 = Wissen"
1870 H$(12)="  9 = Inverse weergave"
1880 H$(13)=" 10 = 1 en 2 verwisselen"
1890 REM Tweede beeldscherm wissen
1900 FOR N=9216 TO 9126+767
1910 VPOKE N,32
1920 NEXT N
```

```
1930 REM HELP-regels printen
1940 FOR N=1 TO 13
1950 FOR M=1 TO LEN(H$(N))
1960 VPOKE 9215+(N*32+M),ASC(MID$(H$(N),M,1))
1970 NEXT M
1980 NEXT N
1990 REM De 5 sprites uit de DATA-regels laden
2000 FOR N=1 TO 5
2010 S$=SPACE$(32)
2020 FOR M=1 TO 16
2030 READ A$
2040 MID$(S$,M,1)=CHR$(VAL("&b"+LEFT$(A$,8)))
2050 MID$(S$,M+16,1)=CHR$(VAL("&b"+RIGHT$(A$,8)))
2060 NEXT M
2070 SPRITE$(N)=S$
2080 NEXT N
2090 REM Zichtbare start van het programma
2100 CLS
2110 PRINT
2120 PRINT STRING$(30,252);
2130 PRINT
2140 REM Afbeelding van de karakters 33-126
2150 FOR N=33 TO 126
2160 PRINT CHR$(N);
2170 NEXT N
2180 PRINT " ";
2190 REM Afbeelding van de karakters 128-242
2200 FOR N=128 TO 242
2210 PRINT CHR$(N);
2220 NEXT N
2230 PRINT
2240 PRINT
2250 REM Afbeelding van de 2 edit-blokken
2260 FOR N=1 TO 8
2270 PRINT " ";STRING$(8,CHR$(251));" ";STRING$(8,CHR$(251))
2280 NEXT N
2290 REM Afbeelding van 10 tekens als controle op de wijziging
2300 LOCATE 19,15
2310 PRINT CHR$(251);
2320 LOCATE 25,15
2330 PRINT CHR$(251);
2340 FOR N=21 TO 23
2350 FOR M=14 TO 16
2360 LOCATE N,M
2370 PRINT CHR$(251);
2380 NEXT M,N
2390 FOR N=27 TO 29
2400 FOR M=14 TO 16
2410 LOCATE N,M
```

```
2420 PRINT CHR$(251);
2430 NEXT M,N
2440 LOCATE 0,21
2450 PRINT STRING$(30,CHR$(252));
2460 REM De 4 blok-sprites en de edit-sprite op het scherm zetten
2470 PUT SPRITE 5,(6,21),15
2480 PUT SPRITE 1,(13,92),15
2490 PUT SPRITE 2,(51,92),15
2500 PUT SPRITE 3,(13,130),15
2510 PUT SPRITE 4,(51,130),15
2520 REM Beginpunt van karaktercursor-sprite
2530 Z1=6
2540 Z2=21
2550 C1=33
2560 REM Controle op uitlezing toetsenbord
2570 A$=INKEY$
2580 IF A$="" THEN GOTO 2570
2590 REM ESC: van karakterset naar wijzigingsoptie
2600 IF ASC(A$)=27 THEN ED=1:GOTO 3210
2610 REM Cursorbesturing inlezen
2620 IF ASC(A$)=28 AND (Z1<>238) THEN Z1=Z1+8:
PUT SPRITE 5,(Z1,Z2),15:C1=C1+1:GOTO 3120
2630 IF ASC(A$)=29 AND (Z1<>6) THEN Z1=Z1-8:
PUT SPRITE 5,(Z1,Z2),15:C1=C1-1:GOTO 3120
2640 IF ASC(A$)=30 AND (Z1<>21) THEN Z2=Z2-8:
PUT SPRITE 5,(Z1,Z2),15:C1=C1-30:GOTO 3120
2650 IF ASC(A$)=31 AND (Z1<>69) THEN Z2=Z2+8:
PUT SPRITE 5,(Z1,Z2),15:C1=C1+30:GOTO 3120
2660 REM TAB: aangestuurde teken in het blok groot weergeven
2670 IF ASC(A$)=9 THEN IF M1=2 THEN GOTO 2900
 ELSE GOTO 2690 ELSE GOTO 2570
2680 REM Linker blok (M1=1)
2690 FOR N=0 TO 7
2700 A$(N)=BIN$(VPEEK(BASE(7)+(C1*8)+N))
2710 A$(N)=STRING$(8-LEN(A$(N)),"0")+A$(N)
2720 NEXT N
2730 FOR N=0 TO 7
2740 FOR M=0 TO 7
2750 LOCATE M+1,N+12
2760 IF MID$(A$(N),M+1,1)="1" THEN PRINT CHR$(252);
 ELSE PRINT CHR$(251);
2770 A(M+1,N+1)=VAL(MID$(A$(N),M+1,1))
2780 NEXT M
2790 NEXT N
2800 LOCATE 19,15
2810 PRINT CHR$(C1);
2820 FOR N=21 TO 23
2830 FOR M=14 TO 16
2840 LOCATE N,M
```

```
2850 PRINT CHR$(C1);
2860 NEXT M
2870 NEXT N
2880 GOTO 2570
2890 REM Rechter blok (M1=2)
2900 FOR N=0 TO 7
2910 B$(N)=BIN$(VPEEK(BASE(7)+(C1*8)+N))
2920 B$(N)=STRING$(8-LEN(B$(N)),"0")+B$(N)
2930 NEXT N
2940 FOR N=0 TO 7
2950 FOR M=0 TO 7
2960 LOCATE M+10,N+12
2970 IF MID$(B$(N),M+1,1)="1" THEN PRINT CHR$(252);
 ELSE PRINT CHR$(251);
2980 B(M+1,N+1)=VAL(MID$(B$(N),M+1,1))
2990 NEXT M
3000 NEXT N
3010 LOCATE 25,15
3020 PRINT CHR$(C1);
3030 FOR N=27 TO 29
3040 FOR M=14 TO 16
3050 LOCATE N,M
3060 PRINT CHR$(C1);
3070 NEXT M
3080 NEXT N
3090 REM Nummer van karakter als punten en strepen
3100 GOTO 2570
3110 NEXT N
3120 LOCATE 0,0
3130 PRINT SPACE$(23);
3140 LOCATE 0,0
3150 A$=CHR$(254)
3160 IF C1>99 THEN Z=INT(C1/100):A$=A$+STRING$(Z,CHR$(253))
+CHR$(254):C2=C1-(Z*100) ELSE A$=A$+" "+CHR$(254):C2=C1
3170 IF C2>9 THEN Z=INT(C2/10):A$=A$+STRING$(Z,CHR$(253))
+CHR$(254):C3=C2-(Z*10) ELSE A$=A$+" "+CHR$(254):C3=C2
3180 IF C3>0 THEN Z=C3:A$=A$+STRING$(Z,CHR$(253))+CHR$(254)
 ELSE A$=A$+" "+CHR$(254)
3190 PRINT A$;
3200 GOTO 2570
3210 REM Werk als editor met vergrote karakterweergave
3220 IF M1=2 THEN GOTO 3420
3230 REM Werk in linker blok (M1=1)
3240 M5=14
3250 M6=93
3260 M7=1
3270 M8=1
3280 PUT SPRITE 5,(M5,M6),15
3290 A$=INKEY$:IF A$="" THEN GOTO 3290
```

```
3300 REM ENTER: veranderingen opslaan onder het karakternummer
3310 IF ASC(A$)=13 THEN FOR N=0 TO 7:FOR M=1 TO 8:Q$=Q$+RIGHT$
(STR$(A(M,N+1)),1):NEXT M:VPOKE C1*8+N,VAL("&b"+Q$):Q$="":NEXT N
3320 REM Spatiebalk: punt zetten/wissen
3330 IF A$=" " THEN LOCATE 0+M7,11+M8:IF A(M7,M8)=1 THEN PRINT
 CHR$(251);:A(M7,M8)=0 ELSE PRINT CHR$(252);:A(M7,M8)=1
3340 REM ESC: van wijzigingsoptie naar karakterset
3350 IF ASC(A$)=27 THEN ED=0:PUT SPRITE 5,(Z1,Z2),15:GOTO 2570
3360 REM Cursorbesturing inlezen
3370 IF ASC(A$)=28 AND M7<>8 THEN M5=M5+8:M7=M7+1:
PUT SPRITE 5,(M5,M6),15:GOTO 3290
3380 IF ASC(A$)=29 AND M7<>1 THEN M5=M5-8:M7=M7-1:
PUT SPRITE 5,(M5,M6),15:GOTO 3290
3390 IF ASC(A$)=31 AND M8<>8 THEN M6=M6+8:M8=M8+1:
PUT SPRITE 5,(M5,M6),15:GOTO 3290
3400 IF ASC(A$)=30 AND M8<>1 THEN M6=M6-8:M8=M8-1:
PUT SPRITE 5,(M5,M6),15:GOTO 3290 ELSE GOTO 3290
3410 REM Werk in rechter blok (M1=2)
3420 M5=86
3430 M6=93
3440 M7=1
3450 M8=1
3460 PUT SPRITE 5,(M5,M6),15
3470 A$=INKEY$
3480 IF A$="" THEN GOTO 3470
3490 REM ENTER: veranderingen opslaan onder het karakternummer
3500 IF ASC(A$)=13 THEN FOR N=0 TO 7:FOR M=1 TO 8:Q$=Q$+RIGHT$
(STR$(B(M,N+1)),1):NEXT M:VPOKE C1*8+N,VAL("&b"+Q$):Q$="":NEXT N
3510 REM Spatiebalk: punt zetten/wissen
3520 IF A$=" " THEN LOCATE 9+M7,11+M8:IF B(M7,M8)=1 THEN PRINT
 CHR$(251);:B(M7,M8)=0 ELSE PRINT CHR$(252);:B(M7,M8)=1
3530 REM ESC: van wijzigingsoptie naar karakterset
3540 IF ASC(A$)=27 THEN ED=0:PUT SPRITE 5,(Z1,Z2),15:GOTO 2570
3560 IF ASC(A$)=28 AND M7<>8 THEN M5=M5+8:M7=M7+1:
PUT SPRITE 5,(M5,M6),15:GOTO 3470
3570 IF ASC(A$)=29 AND M7<>1 THEN M5=M5-8:M7=M7-1:
PUT SPRITE 5,(M5,M6),15:GOTO 3470
3580 IF ASC(A$)=31 AND M8<>8 THEN M6=M6+8:M8=M8+1:
PUT SPRITE 5,(M5,M6),15:GOTO 3470
3590 IF ASC(A$)=30 AND M8<>1 THEN M6=M6-8:M8=M8-1:
PUT SPRITE 5,(M5,M6),15:GOTO 3470 ELSE GOTO 3470
3600 REM Functietoets 1: karakterset laden
3610 LOCATE 10,0
3620 OPEN "charac" FOR INPUT AS #1
3630 FOR N=33 TO 249
3640 FOR M=0 TO 7
3650 INPUT #1,A%
3660 VPOKE(BASE(7)+(N*8)+M),A%
3670 NEXT M
```

```
3680 NEXT N
3690 CLOSE
3700 RETURN
3710 REM Functietoets 2: naar linker blok
3720 REM Niet mogelijk als u zich in het edit-blok bevindt (ED=1)
3730 IF ED=1 THEN RETURN
3740 PUT SPRITE 1,(13,92),15
3750 PUT SPRITE 2,(51,92),15
3760 PUT SPRITE 3,(13,130),15
3770 PUT SPRITE 4,(51,130),15
3780 M1=1
3790 RETURN
3800 REM Functietoets 3: spiegeling
3810 IF M1=2 THEN GOTO 4010
3820 REM Werk in linker blok (M1=1)
3830 FOR N=1 TO 8
3840 FOR M=1 TO 8
3850 C(N,M)=A(ABS(N-9),M)
3860 NEXT M
3870 NEXT N
3880 FOR N=1 TO 8
3890 FOR M=1 TO 8
3900 A(N,M)=C(N,M)
3910 NEXT M
3920 NEXT N
3930 FOR N=0 TO 7
3940 FOR M=0 TO 7
3950 LOCATE N+1,M+12
3960 IF A(N+1,M+1)=1 THEN PRINT CHR$(252); ELSE PRINT CHR$(251);
3970 NEXT M
3980 NEXT N
3990 RETURN
4000 REM Werk in rechter blok (M1=2)
4010 FOR N=1 TO 8
4020 FOR M=1 TO 8
4030 C(N,M)=B(ABS(N-9),M)
4040 NEXT M
4050 NEXT N
4060 FOR N=1 TO 8
4070 FOR M=1 TO 8
4080 B(N,M)=C(N,M)
4090 NEXT M
4100 NEXT N
4110 FOR N=0 TO 7
4120 FOR M=0 TO 7
4130 LOCATE N+10,M+12
4140 IF B(N+1,M+1)=1 THEN PRINT CHR$(252); ELSE PRINT CHR$(251);
4150 NEXT M
4160 NEXT N
```

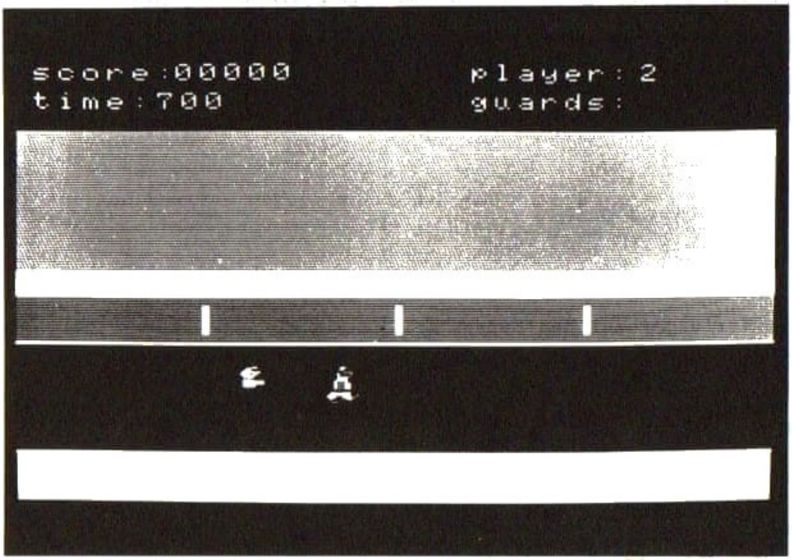
```
4170 RETURN
4180 REM Functietoets 4: rotatie
4190 IF M1=2 THEN GOTO 4390
4200 REM Werk in linker blok
4210 FOR N=1 TO 8
4220 FOR M=1 TO 8
4230 C(N,M)=A(M,9-N)
4240 NEXT M
4250 NEXT N
4260 FOR N=1 TO 8
4270 FOR M=1 TO 8
4280 A(N,M)=C(M,N)
4290 NEXT M
4300 NEXT N
4310 FOR N=0 TO 7
4320 FOR M=0 TO 7
4330 LOCATE N+1,M+12
4340 IF A(N+1,M+1)=1 THEN PRINT CHR$(252); ELSE PRINT CHR$(251);
4350 NEXT M
4360 NEXT N
4370 RETURN
4380 REM Werk in rechter blok (M1=2)
4390 FOR N=1 TO 8
4400 FOR M=1 TO 8
4410 C(N,M)=B(M,9-N)
4420 NEXT M
4430 NEXT N
4440 FOR N=1 TO 8
4450 FOR M=1 TO 8
4460 B(N,M)=C(M,N)
4470 NEXT M
4480 NEXT N
4490 FOR N=0 TO 7
4500 FOR M=0 TO 7
4510 LOCATE N+10,M+12
4520 IF B(N+1,M+1)=1 THEN PRINT CHR$(252); ELSE PRINT CHR$(251);
4530 NEXT M
4540 NEXT N
4550 RETURN
4560 REM Functietoets 5: hulpscherm aan/uit (wisseling tussen
 schermen (VDP(2)) en karaktersets (VDP(4))
4570 IF VDP(2)=6 THEN FOR N=6919 TO 6935 STEP 4: VPOKE N,0:
NEXT N:VDP(4)=5:VDP(2)=9 ELSE VDP(4)=0:VDP(2)=6:FOR N=6919
 TO 6935 STEP 4:VPOKE N,15:NEXT N
4580 RETURN
4590 REM Functietoets 6: karakterset opslaan
4600 LOCATE 10,0
4610 OPEN "charac" FOR OUTPUT AS #1
4620 FOR N=33 TO 249
```

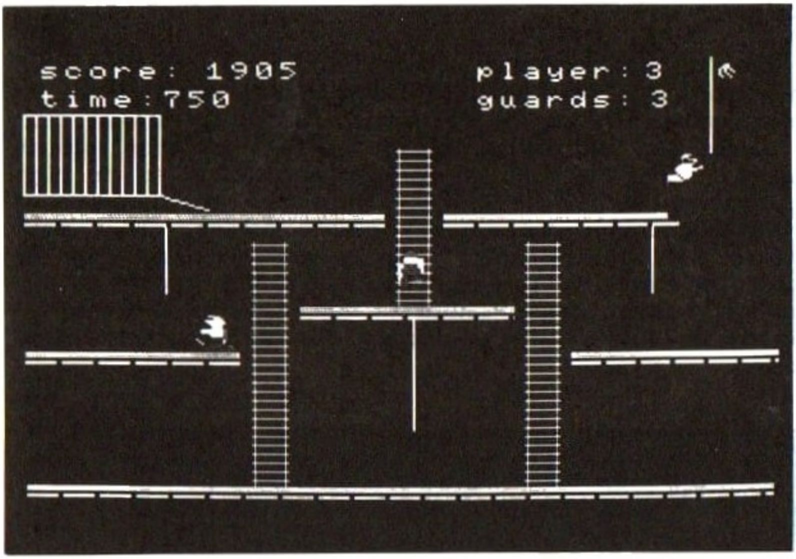
```
4630 FOR M=0 TO 7
4640 A%=VPEEK(BASE(7)+(N*8)+M)
4650 PRINT #1,A%
4660 NEXT M
4670 NEXT N
4680 CLOSE
4690 RETURN
4700 REM Functietoets 7: naar rechter blok
4710 REM Niet mogelijk als u zich in het edit-blok bevindt
4720 IF ED=1 THEN RETURN
4730 PUT SPRITE 1,(85,92),15
4740 PUT SPRITE 2,(123,92),15
4750 PUT SPRITE 3,(85,130),15
4760 PUT SPRITE 4,(123,130),15
4770 M1=2
4780 RETURN
4790 REM Functietoets 8: tekens wissen
4800 IF M1=2 THEN GOTO 4950
4810 REM Werk in linker blok (M1=1)
4820 FOR N=1 TO 8
4830 FOR M=1 TO 8
4840 A(N,M)=0
4850 NEXT M
4860 NEXT N
4870 FOR N=0 TO 7
4880 FOR M=0 TO 7
4890 LOCATE N+1,M+12
4900 PRINT CHR$(251);
4910 NEXT M
4920 NEXT N
4930 RETURN
4940 REM Werk in rechter blok (M1=2)
4950 FOR N=1 TO 8
4960 FOR M=1 TO 8
4970 B(N,M)=0
4980 NEXT M
4990 NEXT N
5000 FOR N=0 TO 7
5010 FOR M=0 TO 7
5020 LOCATE N+10,M+12
5030 PRINT CHR$(251);
5040 NEXT M
5050 NEXT N
5060 RETURN
5070 REM Functietoets 9: tekens invers weergeven
5080 IF M1=2 THEN GOTO 5230
5090 REM Werk in linker blok (M1=1)
5100 FOR N=1 TO 8
5110 FOR M=1 TO 8
```

```
5120 IF A(N,M)=1 THEN A(N,M)=0 ELSE A(N,M)=1
5130 NEXT M
5140 NEXT N
5150 FOR N=0 TO 7
5160 FOR M=0 TO 7
5170 LOCATE N+1,M+12
5180 IF A(N+1,M+1)=0 THEN PRINT CHR$(251); ELSE PRINT CHR$(252);
5190 NEXT M
5200 NEXT N
5210 RETURN
5220 REM Werk in rechter blok (M1=2)
5230 FOR N=1 TO 8
5240 FOR M=1 TO 8
5250 IF B(N,M)=1 THEN B(N,M)=0 ELSE B(N,M)=1
5260 NEXT M
5270 NEXT N
5280 FOR N=0 TO 7
5290 FOR M=0 TO 7
5300 LOCATE N+10,M+12
5310 IF B(N+1,M+1)=0 THEN PRINT CHR$(251); ELSE PRINT CHR$(252);
5320 NEXT M
5330 NEXT N
5340 RETURN
5350 REM Functietoets 10: inhoud van blok 1 en 2 uitwisselen
5360 FOR N=1 TO 8
5370 FOR M=1 TO 8
5380 C(N,M)=A(N,M)
5390 A(N,M)=B(N,M)
5400 B(N,M)=C(N,M)
5410 NEXT M
5420 NEXT N
5430 FOR N=0 TO 7
5440 FOR M=0 TO 7
5450 LOCATE N+1,M+12
5460 IF A(N+1,M+1)=1 THEN PRINT CHR$(252); ELSE PRINT CHR$(251);
5470 LOCATE N+10,M+12
5480 IF B(N+1,M+1)=1 THEN PRINT CHR$(252); ELSE PRINT CHR$(251);
5490 NEXT M
5500 NEXT N
5510 RETURN
5520 REM Bij afbreken van programma alles netjes achterlaten
5530 VDP(2)=6
5540 VDP(4)=0
5550 CLS
5560 SCREEN,3
5570 END
```

9.15 Freddie

Het aapje Freddie is uitgebroken uit de dierentuin en op de vlucht voor zijn verzorgers. Eerst moet hij zich zo snel mogelijk uit de voeten maken omdat een van de verzorgers hem op de hielen zit. Aan het einde van de straat krijgt u het volgende scherm voorgeschoteld. Freddie moet nu bananen verzamelen en er zijn achtervolgers mee bekogelen. Is het aantal bewakers gereduceerd tot nul, dan verschijnt opnieuw het eerste scherm. Weer moet Freddie maken dat hij weg komt, maar nu nog sneller, want de bewaker heeft de smaak te pakken. Het aantal achtervolgers dat vervolgens moet worden uitgeschakeld is groter geworden zodat het maar de vraag is of Freddie die opdracht binnen de gestelde tijd kan vervullen.






```
10 REM Freddie
20 REM MSX grafiek en geluid
30 REM Rainer Lüers en Olaf Otis
40 DEFINT C-Y,A
50 SCREEN,,0
60 OPEN "grp:"FOR OUTPUT AS #1
70 DIM M(20)
80 DIM SD(8)
90 COLOR 10,1,1
100 SCREEN 2
110 LINE(30,40)-(220,130),15,B
120 DRAW "bm 95,85"
130 PRINT #1,"FREDDIE"
140 DRAW "bm 55,150"
150 PRINT #1,"Wilt u informatie"
160 DRAW "bm 95,170"
170 PRINT #1,"Ja/Nee"
180 A$=INKEY$
190 IF A$="J" OR A$="j" THEN GOSUB 6920 ELSE IF A$="N"
 OR A$="n" THEN GOTO 210
200 GOTO 180
210 SCREEN 2
220 DRAW "bm 40,10"
230 PRINT #1,"Speelt u met toetsen (0)"
240 DRAW "bm 40,30"
250 PRINT #1,"of met een joystick (1)"
260 DRAW "bm 97,50"
```

```
270 PRINT #1,"(0)/(1)"
280 DRAW"bm 2,6"
290 A$=INKEY$
300 IF A$="1" THEN AZ=1:GOTO 330
310 IF A$="0" THEN AZ=0:GOTO 330
320 GOTO 290
330 SCREEN 2,2
340 FOR A=1 TO 30 STEP 8
350 LINE(30,40)-(220,130),15,B
360 DRAW"bm 48,80"
370 COLOR 10
380 PRINT #1,"Het programma wordt"
390 DRAW "bm 70,100"
400 PRINT #1,"nu ingelezen!"
410 COLOR 14
420 FOR L=1 TO 20
430 READ M(L)
440 NEXT L
450 DATA 1,4,7,9,11,12,13,14,15,15,15,15,14,13,12,11,9,7,4,0
460 FOR CB=1 TO 8
470 READ SD(CB)
480 NEXT CB
490 DATA 5,5,4,4,3,3,4,4
500 CB=0
510 REM Definiëring van sprites
520 S$=SPACE$(32)
530 FOR Q=0 TO 13
540 FOR I=1 TO 16
550 READ A$
560 MID$(S$,I,1)=CHR$(VAL("&B"+LEFT$(A$,8)))
570 MID$(S$,I+16)=CHR$(VAL("&B"+RIGHT$(A$,8)))
580 NEXT I
590 SPRITE$(Q)=S$
600 NEXT Q
610 RESTORE 1920
620 FOR T=14 TO 25
630 FOR N=1 TO 16
640 READ A$
650 B$=SPACE$(16)
660 FOR U=1 TO LEN(A$)
670 MID$(B$,LEN(A$)+1-U,1)=MID$(A$,U,1)
680 NEXT U
690 A$=B$
700 MID$(S$,N,1)=CHR$(VAL("&B"+LEFT$(A$,8)))
710 MID$(S$,N+16)=CHR$(VAL("&B"+RIGHT$(A$,8)))
720 NEXT N
730 SPRITE$(T)=S$
740 NEXT T
750 REM Weergave
```

```
760 RT=0
770 CV=4
780 ZX=0
790 FA=2
800 ZE=1
810 F$="1"
820 DRAW "bm 52,2"
830 PRINT #1,"00000"
840 DRAW "bm 208,2"
850 PRINT #1,"4"
860 DRAW "bm  6,2"
870 PRINT #1,"score:           speler:"
880 COLOR 1
890 DRAW "bm 80,14"
900 PRINT#1,"   "
910 COLOR 14
920 DRAW "bm  6,14"
930 PRINT #1,"tijd:700         bewakers:"
940 COLOR 1
950 DRAW "bm 204,14"
960 PRINT #1,"  "
970 COLOR 14
980 DRAW "bm 202,14"
990 PRINT #1,FA
1000 IF F$="1" THEN GOTO 5760 ELSE 1010
1010 REM Scherm 2
1020 FOR A=0 TO 255 STEP 12
1030 LINE(A,190)-(A+10,191),13,BF
1040 NEXT A
1050 LINE(0,186)-(255,188),12,BF
1060 FOR A=0 TO 60 STEP 12
1070 LINE(A,130)-(A+10,131),13,BF
1080 LINE(A+90,110)-(A+100,111),13,BF
1090 LINE(A+182,130)-(A+192,131),13,BF
1100 NEXT A
1110 LINE(0,126)-(70,128),12,BF
1120 LINE(90,106)-(162,108),12,BF
1130 LINE(162,110)-(162,111),13
1140 LINE(182,126)-(255,128),12,BF
1150 LINE(254,130)-(255,131),13,BF
1160 FOR A=0 TO 110 STEP 12
1170 LINE(A,70)-(A+10,71),13,BF
1180 LINE(138+A,70)-(148+A,71),13,BF
1190 NEXT A
1200 LINE(220,70)-(255,72),1,BF
1210 LINE(0,66)-(118,68),12,BF
1220 LINE(138,66)-(215,68),12,BF
1230 FOR A=1 TO 108 STEP 4
1240 LINE(74,78+A)-(86,78+A),5
```

```
1250 LINE(166,78+A)-(178,78+A),5
1260 NEXT A
1270 FOR A=1 TO 68 STEP 4
1280 LINE(122,38+A)-(134,38+A),5
1290 NEXT A
1300 LINE(123,38)-(123,105),5
1310 LINE(133,38)-(133,105),5
1320 LINE(75,78)-(75,185),5
1330 LINE(85,78)-(85,185),5
1340 LINE(167,78)-(167,185),5
1350 LINE(177,78)-(177,185),5
1360 LINE(128,112)-(128,160),14
1370 LINE(46,72)-(46,100),14
1380 LINE(210,72)-(210,100),14
1390 LINE(230,0)-(230,40),14
1400 FOR A=0 TO 44 STEP 4
1410 LINE(A,24)-(A,58),14
1420 NEXT A
1430 LINE(0,24)-(44,58),14,B
1440 LINE(44,58)-(60,64),14
1450 PUT SPRITE 5,(230,2),10,13
1460 REM spelverloop
1470 DRAW"bm186,120f5e5"
1480 X=201
1490 Y=169
1500 U=0
1510 O=60
1520 P=49
1530 A$="R"
1540 Q=4
1550 I=0
1560 L=20
1570 RI=1
1580 C=13
1590 S=2
1600 S1=4
1610 A1=180
1620 I1=0
1630 GOSUB 4770
1640 INTERVAL ON
1650 ON INTERVAL=18 GOSUB 4510
1660 IF I1=0 AND Y=109 AND X<194 AND X>174 THEN GOSUB 6670
1670 IF I1=13 AND Y=109 AND X<66 AND X>50 THEN GOSUB 6670
1680 IF STICK(AZ)=3 THEN C=0:D=4:GOSUB 4670
1690 IF STICK(AZ)=7 THEN C=13:D=-4:GOSUB 4670
1700 IF STICK(AZ)=1 THEN Z=-2:GOSUB 4810
1710 IF STICK(AZ)=5 THEN Z=2:GOSUB 4810
1720 IF POINT(X,Y+17)=12 AND STICK(AZ)=2 THEN C=0:RI=1:GOSUB 5000
1730 IF POINT(X,Y+17)=12 AND STICK(AZ)=8 THEN C=13:RI=-1:GOSUB 5000
```

```
1740 IF T$="J" AND STRIG(AZ)=-1 THEN T$="N":GOSUB 5280
1750 GOTO 1660
1760 DATA 0000000000000000
1770 DATA 0000000000000000
1780 DATA 0000000000000000
1790 DATA 0000000000000000
1800 DATA 0000000110000000
1810 DATA 0000000110000000
1820 DATA 0000000110000000
1830 DATA 0000000110000000
1840 DATA 0000000110000000
1850 DATA 0000000110000000
1860 DATA 0000000110000000
1870 DATA 0000000110000000
1880 DATA 0000000110000000
1890 DATA 0000000110000000
1900 DATA 0000000110000000
1910 DATA 0000000110000000
1920 '
1930 DATA 0000001110000000
1940 DATA 0000011100000000
1950 DATA 0000010111100000
1960 DATA 0000011001100000
1970 DATA 0000011111000000
1980 DATA 0000000000000000
1990 DATA 0000011100000000
2000 DATA 0000011100000000
2010 DATA 0000011100000000
2020 DATA 0000011100000000
2030 DATA 0000011000000000
2040 DATA 0000111000000000
2050 DATA 0010000000011000
2060 DATA 0111000001110000
2070 DATA 0011000011100000
2080 DATA 0001110001000000
2090 '
2100 DATA 0000001110000000
2110 DATA 0000011100000000
2120 DATA 0000010111100000
2130 DATA 0000011001100000
2140 DATA 0000011111000000
2150 DATA 0000000000000000
2160 DATA 0000011100000000
2170 DATA 0000011100000000
2180 DATA 0000011100000000
2190 DATA 0000011100000000
2200 DATA 0000011000000000
2210 DATA 0000111000000000
2220 DATA 0000000000000000
```

```
2230 DATA 0000000000000000
2240 DATA 0000111000000000
2250 DATA 0000111111000000
2260 '
2270 DATA 0000001100000000
2280 DATA 0000000000000000
2290 DATA 0000000000000000
2300 DATA 0000010000000000
2310 DATA 0000000000000000
2320 DATA 0000011110000000
2330 DATA 0000100010000000
2340 DATA 0000100011000000
2350 DATA 0000100011000000
2360 DATA 0000100011000000
2370 DATA 0000000000000000
2380 DATA 0001100111000000
2390 DATA 0011111111000000
2400 DATA 0001100111000000
2410 DATA 0000000000000000
2420 DATA 0000000000000000
2430 '
2440 DATA 0000001100000000
2450 DATA 0000000000000000
2460 DATA 0000000000000000
2470 DATA 0000010000000000
2480 DATA 0000000000000000
2490 DATA 0000011110000000
2500 DATA 0000100010000000
2510 DATA 0000100011000000
2520 DATA 0000100011000000
2530 DATA 0000100011000000
2540 DATA 0000000000000000
2550 DATA 0000100111000000
2560 DATA 0000111110000000
2570 DATA 0000111000000000
2580 DATA 0000000000000000
2590 DATA 0000000000000000
2600 '
2610 DATA 0000011111000000
2620 DATA 0000111100000000
2630 DATA 0000111111100000
2640 DATA 0000111111000000
2650 DATA 0000011110000000
2660 DATA 0000000000000000
2670 DATA 0000011111110000
2680 DATA 0000011111110000
2690 DATA 0000011110000000
2700 DATA 0000000000000000
2710 DATA 0000000000000000
```

```
2720 DATA 0000000000000000
2730 DATA 0000000000000000
2740 DATA 0000000000000000
2750 DATA 0000000000000000
2760 DATA 0000000000000000
2770 '
2780 DATA 0000011100000000
2790 DATA 0000110000001000
2800 DATA 0000100000001000
2810 DATA 0000001110001000
2820 DATA 0000000000001000
2830 DATA 0000111110001000
2840 DATA 0001100000001000
2850 DATA 0001100000001000
2860 DATA 0001100001010000
2870 DATA 0001111111000000
2880 DATA 0001111111000000
2890 DATA 0000000000000000
2900 DATA 0010111111000000
2910 DATA 0111011111101000
2920 DATA 1111001111110000
2930 DATA 0011100011100000
2940 '
2950 DATA 0000011100000000
2960 DATA 0000110000000001
2970 DATA 0000100000000001
2980 DATA 0000001110000001
2990 DATA 0000000000000010
3000 DATA 0000111110000010
3010 DATA 0001100000000100
3020 DATA 0001100000001000
3030 DATA 0001100001010000
3040 DATA 0001111111000000
3050 DATA 0001111111000000
3060 DATA 0000000000000000
3070 DATA 0001111111000000
3080 DATA 0001111100000000
3090 DATA 0001110000000000
3100 DATA 0001111100000000
3110 '
3120 DATA 0000000000000000
3130 DATA 0000000000000000
3140 DATA 0000010001000000
3150 DATA 0000011111110000
3160 DATA 0000011111110000
3170 DATA 0000001110110000
3180 DATA 0001100000110000
3190 DATA 0001100000110000
3200 DATA 0001100000110000
```

```
3210 DATA 0001100000000000
3220 DATA 0001100000000000
3230 DATA 0001100000000000
3240 DATA 0000000000000000
3250 DATA 0000000000000000
3260 DATA 0000000000000000
3270 DATA 0000000000000000
3280 '
3290 DATA 0000001110000000
3300 DATA 0000011111000000
3310 DATA 0000011111000000
3320 DATA 0000000000000000
3330 DATA 0000000000000000
3340 DATA 0000011111000000
3350 DATA 0000011111000000
3360 DATA 0000111111100000
3370 DATA 0000111111100000
3380 DATA 0000111111100000
3390 DATA 0000000000000000
3400 DATA 0000111111100000
3410 DATA 0000111011110000
3420 DATA 0000111000000000
3430 DATA 0000111000000000
3440 DATA 0001111000000000
3450 '
3460 DATA 0000000000000000
3470 DATA 0000011111000000
3480 DATA 0000011111110000
3490 DATA 0000011111110000
3500 DATA 0000011111110000
3510 DATA 0000001110110000
3520 DATA 0001100000110000
3530 DATA 0001100000110000
3540 DATA 0001100000110000
3550 DATA 0001100000000000
3560 DATA 0001100000000000
3570 DATA 0001100011100000
3580 DATA 0001100011110000
3590 DATA 0000000000000000
3600 DATA 0000111000000000
3610 DATA 0001111000000000
3620 '
3630 DATA 0000000011100000
3640 DATA 0000000111000000
3650 DATA 0000000101111000
3660 DATA 0000000100011000
3670 DATA 0000000111110000
3680 DATA 0000000000000110
3690 DATA 0000000111111110
```

```
3700 DATA 0000001111111000
3710 DATA 0000000000000000
3720 DATA 0011000000000000
3730 DATA 0011000000000000
3740 DATA 0011000000000000
3750 DATA 0010000000000000
3760 DATA 0010000000000000
3770 DATA 0010000000000000
3780 DATA 0000000000000000
3790 '
3800 DATA 0000000010000000
3810 DATA 0000000100000000
3820 DATA 0000000000000000
3830 DATA 0000000000000000
3840 DATA 0000000000000000
3850 DATA 0000001111100000
3860 DATA 0000011000000000
3870 DATA 0000010000000000
3880 DATA 0000001111100000
3890 DATA 0000110111100000
3900 DATA 0000111011000000
3910 DATA 0000111100000000
3920 DATA 0000000000000000
3930 DATA 0000000000000000
3940 DATA 0000000000000000
3950 DATA 0000000000000000
3960 '
3970 DATA 0000011000000000
3980 DATA 0000111000000000
3990 DATA 0001010100000000
4000 DATA 0001010010000000
4010 DATA 0001001000000000
4020 DATA 0000100100000000
4030 DATA 0000000000000000
4040 DATA 0000000000000000
4050 DATA 0000000000000000
4060 DATA 0000000000000000
4070 DATA 0000000000000000
4080 DATA 0000000000000000
4090 DATA 0000000000000000
4100 DATA 0000000000000000
4110 DATA 0000000000000000
4120 DATA 0000000000000000
4130 REM Links-rechts-bewakers
4140 GOSUB 5630
4150 O=O+Q
4160 G=G+1
4170 E=(-1)^G
4180 IF E=-1 THEN F=6
```

```
4190 IF E=1 THEN F=7
4200 PUT SPRITE 4,(O,P),11,5+I
4210 PUT SPRITE 3,(O,P),4,F+I
4220 IF Q=4 AND POINT(O,P+17)<>12 THEN A$="T"
4230 IF Q=-4 AND POINT(O+16,P+17)<>12 THEN A$="T"
4240 IF O=<1 THEN O=60:P=49:I=0:Q=4
4250 IF O>238 THEN O=60:P=49:I=0
4260 IF Y=P AND O>X-8 AND O<X+8 THEN GOSUB 6670
4270 RETURN
4280 REM Bewakers voor trappen
4290 GOSUB 5630
4300 IF O>130 THEN J=4 ELSE J=0
4310 P=P+2
4320 G=G+1
4330 E=(-1)^G
4340 IF E=-1 THEN K=0
4350 IF E=1 THEN K=13
4360 PUT SPRITE 4,(O+J,P),11,8+K
4370 PUT SPRITE 3,(O+J,P),4,9+K
4380 IF POINT(8+0,P+17)=12 THEN GOSUB 4420
4390 IF P>Y-16 AND P<Y+16 AND O+J>X-10 AND O+J<X+26
 THEN GOSUB 6670
4400 RETURN
4410 REM Richting
4420 A$="R"
4430 IF X<128 AND P<120 THEN Q=-4:I=13
4440 IF X>128 AND P<120 THEN Q=4:I=0
4450 IF X>160 AND P>120 THEN Q=4:I=0
4460 IF X<80 AND P>120 THEN Q=-4:I=13
4470 IF X<160 AND O>150 AND P>120 THEN Q=-4:I=13
4480 IF X>80 AND O<90 AND P>120 THEN Q=4:I=13
4490 RETURN
4500 REM Interval
4510 ZX=ZX+1
4520 IF ZX MOD 50=0 THEN LINE(46,14)-(66,22),1,BF:
DRAW "bm 46,14":COLOR 14:PRINT #1,USING "###";900-ZX/2
4530 IF ZX=1800 THEN GOSUB 6680
4540 CB=CB+1
4550 IF CB=9 THEN CB=1
4560 SOUND 8,16
4570 SOUND 12,6
4580 SOUND 13,9
4590 SOUND 1,SD(CB)
4600 SOUND 10,16
4610 SOUND 4,2
4620 SOUND 12,5
4630 SOUND 13,8
4640 IF A$="R" THEN GOSUB 4140
4650 IF A$="T" THEN GOSUB 4290
```

```
4660 RETURN
4670 REM Links-rechts-aap
4680 X=X+D
4690 IF X>240 THEN X=X-4
4700 IF X<2 THEN X=X+4
4710 IF POINT(X-5,Y+17)<>12 AND POINT(X+24,Y+17)<>12 THEN
 GOSUB 4920
4720 U=U+1
4730 V=(-1)^U
4740 IF V=-1 THEN S=1
4750 IF V=1 THEN S=2
4760 IF STICK(AZ)<>3 AND STICK(AZ)<>7 THEN S=2
4770 PUT SPRITE 2,(X,Y),6,S+C
4780 PUT SPRITE 1,(X,Y),10,2+S+C
4790 RETURN
4800 REM Aap op trap
4810 IF POINT(X+2,Y)=5 AND POINT(X+12,Y)=5 THEN GOTO 4820 ELSE
 RETURN
4820 IF Z=2 AND POINT(X+8,Y+17)=12 THEN RETURN
4830 Y=Y+Z
4840 U=U+1
4850 V=(-1)^U
4860 IF V=-1 THEN S=0
4870 IF V=1 THEN S=13
4880 PUT SPRITE 2,(X,Y),10,9+S
4890 PUT SPRITE 1,(X,Y),6,10+S
4900 RETURN
4910 REM Val
4920 IF STICK(AZ)=7 THEN X=X-8
4930 IF STICK(AZ)=3 THEN X=X+8
4940 Y=Y+1
4950 PUT SPRITE 2,(X,Y),10,9
4960 PUT SPRITE 2,(X,Y),6,10
4970 IF POINT(X,Y+17)=12 OR POINT(X+16,Y+17)=12 GOTO 1660
4980 GOTO 4940
4990 REM Sprong
5000 FOR L=1 TO 20
5010 PUT SPRITE 1,(X+RI*L,Y-M(L)),10,12+C
5020 PUT SPRITE 2,(X+RI*L,Y-M(L)),6,11+C
5030 IF POINT(X+RI*L+8,Y-M(L))=14 THEN GOSUB 5100
5040 IF X+RI*L+8>0 AND X+RI*L+8<0+16 AND Y-M(L)+16>P
 AND Y-M(L)+16<P+16 THEN GOTO 6680
5050 NEXT L
5060 X=X+RI*20
5070 D=0
5080 GOTO 4670
5090 '
5100 Y=Y-8
5110 PUT SPRITE 2,(X+RI*L,Y-M(L)),10,3+C
```

```
5120 PUT SPRITE 1,(X+RI*L,Y-M(L)),6,1+C
5130 IF STICK(AZ)=5 THEN GOSUB 5190
5140 IF STICK(AZ)=1 THEN Y=Y-1
5150 IF POINT(X+RI*L+8,Y-M(L))<>14 THEN Y=Y+1
5160 IF Y-M(L)<5 THEN BEEP:PUT SPRITE 5,(200,200):T$="J":
GOSUB 5190
5170 GOTO 5110
5180 '
5190 Y=Y+1
5200 PUT SPRITE 1,(X+RI*L,Y-M(L)),10,12+C
5210 PUT SPRITE 2,(X+RI*L,Y-M(L)),6,11+C
5220 IF POINT(X+RI*L,Y-M(L)+17)=12 THEN X=X+RI*8:Y=Y-M(L):
GOTO 5240
5230 GOTO 5190
5240 PUT SPRITE 2,(X,Y),6,2+C
5250 PUT SPRITE 1,(X,Y),10,4+C
5260 GOTO 1660
5270 REM Worp
5280 IF C=0 THEN AS=6
5290 IF C=13 THEN AS=-6
5300 PUT SPRITE 1,(X,Y),10,12+C
5310 PUT SPRITE 2,(X,Y),6,11+C
5320 FOR FG=1 TO 20
5330 PUT SPRITE 5,(X+FG*AS+8,Y-5),10,13
5340 IF X+FG*AS+8>0 AND X+FG*AS+8<0+16 AND Y=P THEN V$="S":
GOSUB 5440
5350 IF X+FG*AS+8>A1 AND X+FG*AS+8<A1+16 AND Y=109 THEN V$="L":
GOSUB 5440
5360 IF X+FG*AS+8>250 THEN 5390
5370 IF X+FG*AS+8<2 THEN 5390
5380 NEXT FG
5390 PUT SPRITE 2,(X,Y),6,S+C
5400 PUT SPRITE 1,(X,Y),10,2+S+C
5410 PUT SPRITE 5,(230,2),10,13
5420 GOTO 1750
5430 '
5440 RT=RT+1
5450 ER=ER+3300
5460 O=60
5470 P=49
5480 BEEP
5490 IF V$="S" THEN O=60:P=49:Q=4:I=0
5500 IF V$="L" THEN A1=180:PUT SPRITE 5,(100,200):
PUT SPRITE 6,(100,200)
5510 COLOR 1
5520 DRAW"bm 204,14"
5530 LINE(204,14)-(220,22),1,BF
5540 IF FA-RT=0 THEN ER=ER+1000:INTERVAL OFF:ZX=0:
RT=0:ZE=ZE+.1:GOTO 5760
```

```
5550 LINE(50,2)-(100,10),1,BF
5560 COLOR 14:DRAW"bm 196,14":PRINT #1,FA-RT
5570 LINE(50,2)-(100,10),1,BF
5580 DRAW "bm 52,2"
5590 PRINT #1,USING"#####";ER
5600 IF ERR>10000 THEN ER=0
5610 GOTO 5390
5620 REM Bewakers 2
5630 A1=A1+S1
5640 Q1=Q1+1
5650 W1=(-1)^Q1
5660 IF W1=-1 THEN F1=6
5670 IF W1=1 THEN F1=7
5680 PUT SPRITE 7,(A1,109),11,5+I1
5690 PUT SPRITE 6,(A1,109),4,F1+I1
5700 IF A1>240 THEN A1=6
5710 IF A1>170 AND A1<180 THEN S1=4:I1=0:
DRAW"c14bm186,120f5e5":DRAW"c1bm56,120f5e5"
5720 IF A1>60 AND A1<70 THEN S1=-4:I1=13:
DRAW"c14bm56,120f5e5":DRAW"c1bm186,120f5e5"
5730 IF A1<2 THEN A1=236:S1=-4:I1=13
5740 IF A1>X-8 AND Y=109 AND A1<8+X THEN GOSUB 6670
5750 RETURN
5760 REM Scherm 1
5770 FOR A=1 TO 7
5780 PUT SPRITE A,(100,200)
5790 NEXT A
5800 LINE(0,40)-(250,190),1,BF
5810 COLOR 1
5820 LINE(204,14)-(220,22),1,BF
5830 COLOR 14
5840 LINE(0,24)-(60,40),1
5850 LINE(230,0)-(230,40),1
5860 LINE(0,24)-(44,40),1,BF
5870 LINE(0,120)-(256,170),1,BF
5880 LINE(0,30)-(255,90),5,BF
5890 LINE(0,101)-(255,120),12,BF
5900 LINE(0,90)-(255,100),10,BF
5910 LINE(0,170)-(255,191),2,BF
5920 LINE(0,121)-(255,121),15
5930 LINE(0,169)-(255,169),15
5940 B=0:K$="N":GOSUB 6110
5950 INTERVAL ON
5960 ON INTERVAL=12 GOSUB 6220
5970 Z=0
5980 IF K$="N" AND STICK(AZ)=3 THEN K$="J":GOSUB 6010
5990 IF K$="J" AND STICK(AZ)=7 THEN K$="N":GOSUB 6010
6000 GOTO 5980
6010 SOUND 9,16
```

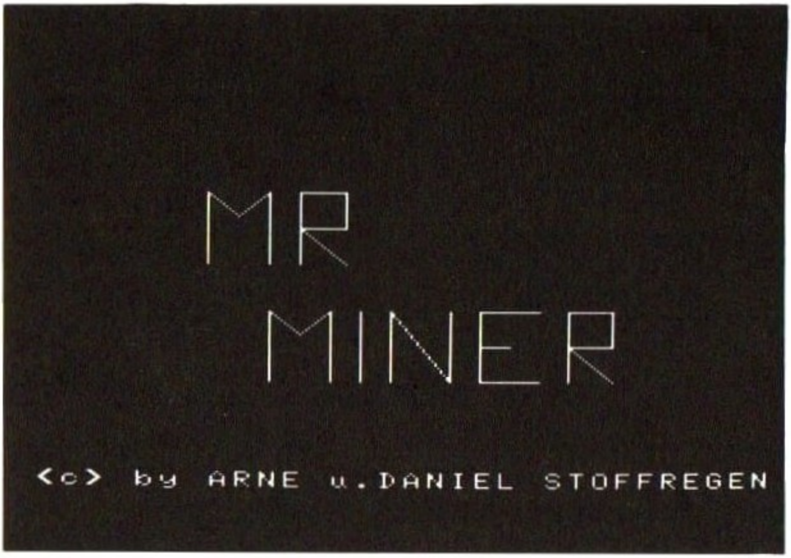
```
6020 SOUND 13,9
6030 SOUND 12,3
6040 SOUND 3,0
6050 SOUND 7,B4
6060 U=U+1
6070 E=(-1)^U
6080 IF E=-1 THEN S=0
6090 IF E=1 THEN S=1
6100 Z=Z+.5
6110 PUT SPRITE 2,(100+Z,130),6,1+S
6120 PUT SPRITE 1,(100+Z,130),10,3+S
6130 A=A-4
6140 IF A=64 THEN A=0
6150 PUT SPRITE 5,(A+50,100),15,0
6160 PUT SPRITE 6,(A+114,100),15,0
6170 PUT SPRITE 7,(A+178,100),15,0
6180 PUT SPRITE 8,(A+246,100),15,0
6190 IF 100+Z>240 THEN GOTO 6320
6200 RETURN
6210 REM Bewakers
6220 RG=RG+1
6230 T=(-1)^RG
6240 IF T=-1 THEN I=6
6250 IF T=1 THEN I=7
6260 B=B+ZE
6270 PUT SPRITE 4,(50+B,130),11,5
6280 PUT SPRITE 3,(50+B,130),4,I
6290 IF 64+B>100+Z THEN GOTO 6540
6300 RETURN
6310 REM Einde van scherm 1
6320 INTERVAL OFF
6330 LINE(0,30)-(255,191),1,BF
6340 COLOR 14,1,1
6350 PUT SPRITE 1,(100,200)
6360 PUT SPRITE 2,(100,200)
6370 PUT SPRITE 3,(100,200)
6380 PUT SPRITE 4,(100,200)
6390 PUT SPRITE 5,(100,200)
6400 PUT SPRITE 6,(100,200)
6410 PUT SPRITE 7,(100,200)
6420 PUT SPRITE 8,(100,200)
6430 ER=ER+(100+Z-50+O)*10
6440 BEEP
6450 COLOR 1
6460 LINE(50,2)-(100,10),1,BF
6470 COLOR 14
6480 DRAW"bm 52,2"
6490 PRINT #1,USING"#####";ER
6500 IF ER>10000 THEN ER=0
```

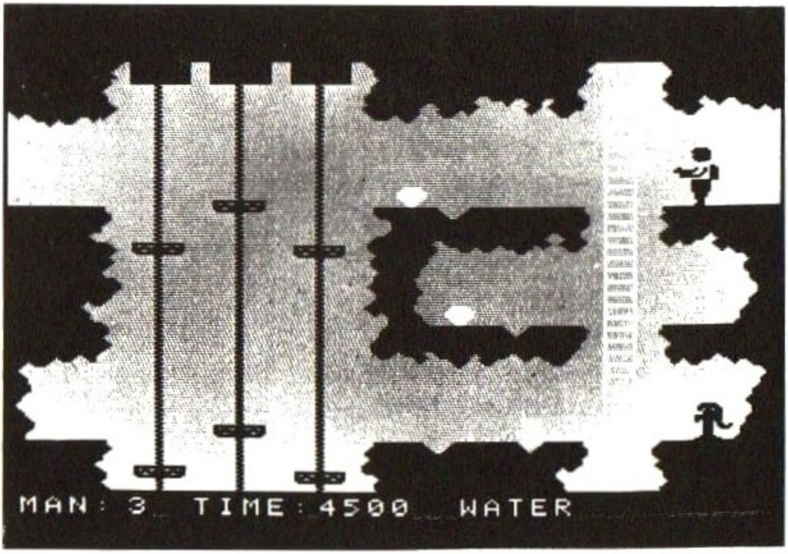
```
6510 FA=FA+1
6520 F$="2"
6530 GOTO 860
6540 REM Aapje minder
6550 Z=0
6560 B=0
6570 CV=CV-1
6580 BEEP
6590 LINE(201,2)-(220,10),1,BF
6600 COLOR 14
6610 DRAW"bm 200,2"
6620 PRINT #1,CV
6630 IF CV=0 THEN GOSUB 6800
6640 FOR R=1 TO 500
6650 NEXT R
6660 RETURN
6670 REM Aapje minder
6680 CV=CV-1
6690 BEEP
6700 LINE(201,2)-(220,10),1,BF
6710 COLOR 14
6720 DRAW"bm 200,2"
6730 PRINT #1,CV
6740 IF CV=0 THEN GOSUB 6800
6750 FOR R=1 TO 500
6760 NEXT R
6770 PUT SPRITE 5,(230,2),10,13
6780 GOTO 1480
6790 REM Einde spel
6800 INTERVAL OFF
6810 SOUND 8,0
6820 SOUND 9,0
6830 SOUND 10,0
6840 LINE(85,80)-(168,110),4,BF
6850 COLOR 15
6860 DRAW"bm 92,90"
6870 PRINT #1,"GAME OVER"
6880 DRAW"bm 92,98"
6890 PRINT #1,"nog eens? J/N"
6900 A$=INKEY$
6910 IF A$="j" OR A$="J" THEN RUN ELSE IF A$="n"
 OR A$="N" THEN END ELSE 6900
6920 SCREEN 2
6930 DRAW "bm 12,10"
6940 PRINT #1,"Scherm 1: door het heen en weer"
6950 DRAW "bm 76,18"
6960 PRINT #1,"bewegen van de joy-"
6970 DRAW "bm 76,26"
6980 PRINT #1,"stick of de cursor-"
```

```
6990 DRAW "bm 76,34"
7000 PRINT #1,"toetsen beweegt de"
7010 DRAW "bm 76,42"
7020 PRINT #1,"aap vooruit om aan de"
7030 DRAW "bm 76,50"
7040 PRINT #1,"bewaker te ontkomen"
7050 DRAW "bm 12,66"
7060 PRINT #1,"Scherm 2: het doel van dit spel"
7070 DRAW "bm 76,74"
7080 PRINT #1,"is het verzamelen van"
7090 DRAW "bm 76,82"
7100 PRINT #1,"bananen en daarmee de"
7110 DRAW "bm 76,90"
7120 PRINT #1,"wachter te bekogelen"
7130 DRAW "bm 76,106"
7140 PRINT #1,"Springen kun je door"
7150 DRAW "bm 76,114"
7160 PRINT #1,"twee cursortoetsen in"
7170 DRAW "bm 76,122"
7180 PRINT #1,"te drukken (b.v. omhoog"
7190 DRAW "bm 76,130"
7200 PRINT #1,"en rechts wordt een"
7210 DRAW "bm 76,138"
7220 PRINT #1,"sprong naar rechts)"
7230 DRAW "bm 30,170"
7240 PRINT #1,"Druk op de spatiebalk"
7250 IF STRIG(0)=-1 THEN GOTO 210 ELSE 7250
```

9.16 Mr. Miner

De mijnwerker in dit computerspelletje kan naar links en rechts bewegen en op de ladder natuurlijk ook van boven naar beneden. Heeft hij water gehaald, dan kan hij nog schieten ook. Om Mr. Miner te laten springen moet u de joystick in de richting noordwest of noordoost drukken. Met de cursorbesturing bereikt u hetzelfde effect door NAAR BOVEN en LINKS of NAAR BOVEN en RECHTS tegelijkertijd in te drukken. Met de drukknop op de joystick of de spatiebalk op het toetsenbord kunt u schieten. In het spelletje komt een vleermuis voor die een ei legt dat, eenmaal op de grond beland, verandert in een diamant. Voor een diamant krijgt u 1000 punten (tijdseenheden) en voor de diamant linksonder zelfs 2000. Het is de bedoeling dat u het mijnwerkertje precies onder zo'n edelsteen dirigeert en vervolgens de joystick naar beneden drukt. De klok begint bij 5000 terug te tellen en wanneer hij bij 0 is, bent u een mijnwerkertje armer. Door diamanten te verzamelen kunt u zijn leven behoorlijk rekken. Om rechtsonder bij de pomp water te tanken drukt u op de rode knop. Bij "WATER" verschijnt dan een blauwe streep die natuurlijk bij elk schot korter wordt. Het is het beste tijdens het spel een aantal malen water te halen. De gevulde tank op de rug van Mr. Miner stelt hem in staat van zijn waterpistool gebruik te maken. Daarmee maakt hij de spoken onschadelijk die hij onderweg tegenkomt. Hij mag er niet mee in aanraking komen, anders is het met hem gedaan. Een dood spook levert 100 punten op. De gang, rechtsboven in beeld, leidt helaas niet naar een ander scherm en het geluid laat wel wat te wensen over. Hopelijk is dat voor u nu juist een uitdaging om met de tot nu toe opgedane kennis eens flink aan de slag te gaan.


MR
MINER
<c> by ARNE u.DANIEL STOFFREGEN


MAN: 3 TIME:4500 WATER

```
10 REM Mr. Miner
20 REM MSX grafiek en geluid
30 REM Rainer Lüers & Arne en Daniel Stoffregen
40 OPEN "grp:"FOR OUTPUT AS #1
50 GOSUB 6820
60 CLEAR 4000
70 OPEN "grp:"FOR OUTPUT AS #1
80 DEFINT A-Z
90 ON STRIG GOSUB 6360,6360,6360
100 DIM SP$(35),B(5),C(5),H$(20),H(20)
110 DATA 5,7,6,7,15,17,16,17
120 REM Loopvlakken
130 FOR T=0 TO 7
140 READ F(T)
150 NEXT T
160 VFL=5
170 GOSUB 6110
180 DRAW "a0"
190 REM Sprongoperatoren (+9,-4 enz.)
200 DATA +9,-4,+9,-2,+9,+2
210 RESTORE 200
220 FOR T=1 TO 3
230 READ B(T),C(T)
240 NEXT T
250 SCREEN 2,2
260 GOSUB 2120 :'sprites
270 FOR T=1 TO 31
280 SPRITE$(T)=SP$(T)
290 NEXT T
300 Q$="r15d3l1d1l6d1l1u1l6u1l1u3
d3r1e2f2e2r1d1r1f1e2f2d1l6u2l1d2"
310 DATA 9,10,7,8,3,4
320 REM Geluid inlezen
330 RESTORE 310
340 FOR T=1 TO 6
350 READ S(T)
360 NEXT T
370 REM Tijd voor de afstand definiëren
380 ON INTERVAL=100 GOSUB 6790
390 REM Alle variabelen op beginwaarde
400 A3=160
410 A5=96
420 A4=106
430 A6=76
440 M=4
450 O=194
460 P=-1
470 F=240
480 REM Begincoördinaten voor mijnwerkertje
```

```
490 X=10
500 Y=20
510 N=0
520 A1=10
530 A2=0
540 L=1
550 L1=0
560 BQ=0
570 WA$="LEEG"
580 LINE (190,182)-(256,192),1,BF
590 TIJD=5000
600 MN=INT(RND(-TIME)*5)+1
610 EI$=""
620 OA=25
630 PUT SPRITE 29,(0,0),0,31
640 X10=240
650 LE$=""
660 REM Geluid definiëren
670 SOUND 7,&B00110100
680 SOUND 1,13
690 SOUND 12,2
700 SOUND 11,55
710 SOUND 8,16
720 SOUND 6,245
730 SOUND 3,6
740 SOUND 9,8
750 IF LI$="N" THEN LI$="":LINE (39,9)-(109,180),3,BF:JJK$="J":
GOSUB 5410
760 PUT SPRITE 27,(0,0),0,31
770 STRIG(1) ON
780 STRIG(0) ON
790 REM Interval starten
800 INTERVAL ON
810 REM Diamanten op verschillende plaatsen laten verschijnen
820 DRAW"bm15,159;"+DI$
830 P$(1)="J"
840 DRAW"bm146,109;"+DI$
850 P$(3)="J"
860 DRAW"bm130,59;"+DI$
870 P$(5)="J"
880 REM Hoofdprogramma
890 A=STICK(0)
900 IF A=0 THEN GOSUB 1560 ELSE IF FA$="N" THEN ON A GOSUB
1670,1440,1450,1430,1800,1430,1500,1550
910 REM Richting (links of rechts)
920 IF R$="1" THEN I=9 ELSE I=0
930 IF J$<>"" THEN 950 ELSE GOSUB 1990
940 REM Liften stoppen
950 IF LI$="N" THEN A2=A2-4:A5=A5-4:GOTO 1020 ELSE
```

```
A1=A1+2:A2=A2-4:A3=A3+3:A4=A4+2:A5=A5-4:A6=A6+3
960 IF A1>192-12 THEN A1=0 ELSE IF A2<0 THEN A2=192-12 ELSE IF
A3>192-12 THEN A3=0 ELSE IF A4>192-12 THEN A4=0 ELSE IF A5<0 THEN
A5=192-12 ELSE IF A6>192-12 THEN A6=0
970 REM Liften plaatsen
980 PUT SPRITE 10,(40,A1),1,24
990 PUT SPRITE 12,(92,A3),1,24
1000 PUT SPRITE 13,(40,A4),1,24
1010 PUT SPRITE 15,(92,A6),1,24
1020 IF LI$="N" THEN PUT SPRITE 27,(X10,22),1,ER ELSE 1130
1030 REM Vogel van rechts naar links laten vliegen
1040 X10=X10-5
1050 IF X10<=0 THEN X10=240:OA=25
1060 REM Vleugels bewegen
1070 RE=RE+1
1080 IF RE/4<>INT(RE/4) THEN IF ER=25 THEN ER=26 ELSE ER=25
1090 IF EI$="J" THEN 1100 ELSE IF X10<PX(MN) AND X10+20>PX(MN)
AND P$(MN)="N" THEN EI$="J" ELSE MN=INT(RND(-TIME)*5)+1:GOTO 1130
1100 PUT SPRITE 29,(PX(MN),OA),15,27
1110 OA=OA+5
1120 IF OA>=PY(MN) THEN EI$="":FG$=STR$(PX(MN)):PUT SPRITE
29,(0,0),0,31:GF$=STR$(PY(MN)):DRAW"bm"+FG$+","+GF$+DI$:P$(MN)="J"
1130 PUT SPRITE 11,(66,A2),1,24
1140 PUT SPRITE 14,(66,A5),1,24
1150 IF LI$="N" THEN IF A2<0 THEN A2=180
1160 IF LI$="N" THEN IF A5<0 THEN A5=180
1170 PUT SPRITE 25,(O,P),15,M
1180 IF X>30 AND X+9<56 THEN GOSUB 2050
1190 IF X>56 AND X+9<82 THEN GOSUB 2070
1200 IF X>82 AND X+9<108 THEN GOSUB 2090
1210 REM Loopkleur instellen bij laddersprong
1220 IF X>185 AND X+16<222 THEN V=11 ELSE V=12
1230 REM Spook bewegen
1240 IF H$(L)="L" THEN O=O-10 ELSE IF H$(L)="R" THEN O=O+10 ELSE
 IF H$(L)="O" THEN P=P-10 ELSE IF H$(L)="U" THEN P=P+10
1250 L1=L1+1
1260 IF L1>H(L) THEN L1=1:L=L+1:IF H$(L)="E" THEN
 O=194:P=0:L=1:L1=0
1270 IF M=4 THEN M=14 ELSE M=4
1280 REM Sprong naar rechts
1290 IF J$="R" THEN X=X+B(J1):Y=Y+C(J1):J1=J1+1:IF J1=3 THEN
 J$="":J1=0
1300 IF SCHOT=1 THEN GOSUB 6380
1310 REM Sprong naar links
1320 IF J$="L" THEN X=X-B(J2):Y=Y+C(J2):J2=J2+1:IF J2=3 THEN
 J$="":J2=0
1330 IF O>=X-4 AND O<=X+6 AND P>=Y-4 AND P<=Y+17 THEN 6270
1340 REM PUT SPRITE (mannetje)
1350 GOSUB 1580
```

```
1360 REM Liften stoppen j/n
1370 BQ=BQ+1
1380 IF BQ=209 THEN GOSUB 6520
1390 DF=DF+1
1400 IF DF=7 THEN DF=1
1410 SOUND 3,S(DF)
1420 GOTO 890
1430 RETURN
1440 IF LE$="J" OR JP$="J" THEN RETURN ELSE
J$="R":R$="r":JP$="J":RETURN
1450 IF LE$="J" THEN LE$="":U=7:R$="r":RETURN ELSE
 SOUND 13,9:R=R+1:IF
R=4 THEN R=0
1460 U=F(R)
1470 X=X+4
1480 R$="r"
1490 RETURN
1500 IF LE$="J" THEN LE$="":U=16:R$="l":RETURN ELSE SOUND
 13,9:K=K+1:IF K=4 THEN K=0
1510 U=F(K+4)
1520 X=X-4
1530 R$="l"
1540 RETURN
1550 IF LE$="J" OR JP$="J" THEN RETURN ELSE
 J$="L":R$="l":JP$="J":RETURN
1560 IF R$="l" THEN U=17 ELSE U=7:RETURN
1570 RETURN
1580 IF R$="l" THEN I=10 ELSE I=0
1590 REM Richting van een sprong bepalen
1600 IF J$="R" THEN U=8 ELSE IF J$="L" THEN U=18
1610 IF LE$="J" THEN RETURN
1620 PUT SPRITE 0,(X,Y),1,1+I
1630 PUT SPRITE 1,(X,Y),8,3+I
1640 PUT SPRITE 2,(X,Y+16),4,U
1650 PUT SPRITE 3,(X,Y),7,2+I:RETURN
1660 REM Omhoog via ladder
1670 IF X<191 OR X>205 THEN RETURN
1680 REM Weergave van de achterzijde
1690 Y=Y-5
1700 X=192
1710 PUT SPRITE 0,(X,Y),1,21
1720 PUT SPRITE 1,(X,Y),8,22
1730 PUT SPRITE 2,(X,Y),7,23
1740 REM Benen bewegen
1750 G=G+1
1760 IF G/2<>INT(G/2) THEN G1=9 ELSE G1=19:G=0
1770 PUT SPRITE 3,(X,Y+16),4,G1
1780 LE$="J"
1790 RETURN
```

```
1800 REM Via ladder naar beneden of diamant oppakken
1810 IF X>191 AND X<205 THEN 1890
1820 IF X>5 AND X<12 AND Y=133 AND P$(1)="J" THEN
 LINE(11,140)-(20,159),3,BF:P$(1)="N":TIJD=TIJD+2000:RETURN
1830 IF X>115 AND X<123 AND Y=133 AND P$(2)="J" THEN
 LINE(121,140)-(130,159),3,BF:P$(2)="N":TIJD=TIJD+1000:RETURN
1840 IF X>136 AND X<143 AND Y=83 AND P$(3)="J" THEN
 LINE(142,90)-(151,109),3,BF:P$(3)="N":TIJD=TIJD+1000:RETURN
1850 IF X>220 AND X<227 AND Y=83 AND P$(4)="J" THEN
 LINE(226,90)-(235,109),3,BF:P$(4)="N":TIJD=TIJD+1000:RETURN
1860 IF X>120 AND X<127 AND Y=33 AND P$(5)="J" THEN
 LINE(126,40)-(135,59),3,BF:P$(5)="N":TIJD=TIJD+1000:RETURN
1870 RETURN
1880 REM Weergave van achterzijde
1890 Y=Y+5
1900 X=192
1910 PUT SPRITE 0,(X,Y),1,21
1920 PUT SPRITE 1,(X,Y),8,22
1930 PUT SPRITE 2,(X,Y),7,23
1940 G=G+1
1950 IF G/2<>INT(G/2) THEN G1=9 ELSE G1=19:G=0
1960 PUT SPRITE 3,(X,Y+16),4,G1
1970 LE$="J"
1980 RETURN
1990 IF POINT(X+8,Y+27)=V THEN FA$="N":N=0:JP$="":RETURN
 ELSE FA$="J"
2000 FOR T=1 TO 5
2010 Y=Y+1
2020 IF POINT(X+8,Y+27)<>V THEN NEXT T ELSE T=6:FA$="N"
2030 IF N=11 THEN 6270 ELSE N=N+1
2040 RETURN
2050 IF Y+26<A1 OR Y+26>A1+5 THEN 2060 ELSE
 Y=A1-26:FA$="N":N=0:JP$="":RETURN
2060 IF Y+26<A4 OR Y+26>A4+5 THEN RETURN ELSE
 Y=A4-26:FA$="N":N=0:JP$="":RETURN
2070 IF Y+26<A2 OR Y+26>A2+9 THEN 2080 ELSE
 Y=A2-26:FA$="N":N=0:JP$="":RETURN
2080 IF Y+26<A5 OR Y+26>A5+9 THEN RETURN ELSE
 Y=A5-26:FA$="N":N=0:JP$="":RETURN
2090 IF Y+26<A3 OR Y+26>A3+7 THEN 2100 ELSE
 Y=A3-26:FA$="N":N=0:JP$="":RETURN
2100 IF Y+26<A6 OR Y+26>A6+7 THEN RETURN ELSE
 Y=A6-26:FA$="N":N=0:JP$="":RETURN
2110 GOTO 2110
2120 S$=SPACE$(32)
2130 RESTORE 2520
2140 FOR T=1 TO 10
2150 FOR N=1 TO 16
2160 READ A$
```

```
2170 MID$(S$,N,1)=CHR$(VAL("&B"+LEFT$(A$,8)))
2180 MID$(S$,N+16)=CHR$(VAL("&B"+RIGHT$(A$,8)))
2190 NEXT N
2200 SP$(T)=S$
2210 S$=SPACE$(32)
2220 NEXT T
2230 RESTORE 2520
2240 GOSUB 5410
2250 S$=SPACE$(32)
2260 FOR T=11 TO 20
2270 FOR N=1 TO 16
2280 READ A$
2290 B$=SPACE$(16)
2300 FOR U=1 TO LEN(A$)
2310 MID$(B$,LEN(A$)+1-U,1)=MID$(A$,U,1)
2320 NEXT U:A$=B$
2330 MID$(S$,N,1)=CHR$(VAL("&B"+LEFT$(A$,8)))
2340 MID$(S$,N+16)=CHR$(VAL("&B"+RIGHT$(A$,8)))
2350 NEXT N
2360 SP$(T)=S$
2370 S$=SPACE$(32)
2380 NEXT T
2390 RESTORE 4220
2400 S$=SPACE$(32)
2410 FOR T=21 TO 27
2420 FOR N=1 TO 16
2430 READ A$
2440 MID$(S$,N,1)=CHR$(VAL("&B"+LEFT$(A$,8)))
2450 MID$(S$,N+16)=CHR$(VAL("&B"+RIGHT$(A$,8)))
2460 NEXT N
2470 SP$(T)=S$
2480 S$=SPACE$(32)
2490 NEXT T
2500 RETURN
2510 REM Sprite 1
2520 DATA 0000000000000000
2530 DATA 0000000000000000
2540 DATA 0000011100000000
2550 DATA 0000111001000000
2560 DATA 0000111010000000
2570 DATA 0000110000000000
2580 DATA 0000100110000000
2590 DATA 0000000000000000
2600 DATA 0000000000000000
2610 DATA 0000000000000000
2620 DATA 0000010010000010
2630 DATA 0000010001111111
2640 DATA 0000001000010100
2650 DATA 0000000111001100
```

```
2660 DATA 0000000000000000
2670 DATA 0000111111000000
2680 REM Sprite 2
2690 DATA 0000000000000000
2700 DATA 0000000000000000
2710 DATA 0000000000000000
2720 DATA 0000000000000000
2730 DATA 0000000000000000
2740 DATA 0000000000000000
2750 DATA 0000000000000000
2760 DATA 0000000000000000
2770 DATA 0000111100000000
2780 DATA 0000111110000000
2790 DATA 0000101101000000
2800 DATA 0000101111000000
2810 DATA 0000110111000000
2820 DATA 0000111000000000
2830 DATA 0000111111000000
2840 DATA 0000000000000000
2850 REM Sprite 3
2860 DATA 0000000000000000
2870 DATA 0000000000000000
2880 DATA 0000000000000000
2890 DATA 0000000110000000
2900 DATA 0000001100000000
2910 DATA 0000001111000000
2920 DATA 0000011000000000
2930 DATA 0000011100000000
2940 DATA 0000000000000000
2950 DATA 0011000000000000
2960 DATA 0111000000000000
2970 DATA 0111000000000000
2980 DATA 0111000000101000
2990 DATA 0111000000110000
3000 DATA 0111000000000000
3010 DATA 0000000000000000
3020 REM Sprite 4
3030 DATA 0000111000000000
3040 DATA 0000011100000000
3050 DATA 0000101010000000
3060 DATA 0000111110000000
3070 DATA 0000011100100000
3080 DATA 0011111111100000
3090 DATA 0010111110000000
3100 DATA 0000111100000000
3110 DATA 0000011100000000
3120 DATA 0000001110000000
3130 DATA 0000000000000000
3140 DATA 0000000000000000
```

```
3150 DATA 0000000000000000
3160 DATA 0000000000000000
3170 DATA 0000000000000000
3180 DATA 0000000000000000
3190 REM Sprite 5
3200 DATA 0000111111100000
3210 DATA 0000111111100000
3220 DATA 0000111111110000
3230 DATA 0000011111111000
3240 DATA 0000011110111100
3250 DATA 0000001100011100
3260 DATA 0000011100111000
3270 DATA 0000011001110000
3280 DATA 0000011000100000
3290 DATA 0000011100100000
3300 DATA 0000000000000000
3310 DATA 0000000000000000
3320 DATA 0000000000000000
3330 DATA 0000000000000000
3340 DATA 0000000000000000
3350 DATA 0000000000000000
3360 REM Sprite 6
3370 DATA 0000111111000000
3380 DATA 0000111111000000
3390 DATA 0000111111000000
3400 DATA 0000011111000000
3410 DATA 0000011111000000
3420 DATA 0111111101100000
3430 DATA 0111110001100000
3440 DATA 0100000011000000
3450 DATA 0000000011000000
3460 DATA 0000000011100000
3470 DATA 0000000000000000
3480 DATA 0000000000000000
3490 DATA 0000000000000000
3500 DATA 0000000000000000
3510 DATA 0000000000000000
3520 DATA 0000000000000000
3530 REM Sprite 7
3540 DATA 0000111111000000
3550 DATA 0000111111000000
3560 DATA 0000111111000000
3570 DATA 0000111111000000
3580 DATA 0000011110000000
3590 DATA 0000011110000000
3600 DATA 0000001110000000
3610 DATA 0000001100000000
3620 DATA 0000001100000000
3630 DATA 0000001110000000
```

```
3640 DATA 0000000000000000
3650 DATA 0000000000000000
3660 DATA 0000000000000000
3670 DATA 0000000000000000
3680 DATA 0000000000000000
3690 DATA 0000000000000000
3700 REM Sprite 8
3710 DATA 0000111111000000
3720 DATA 0001111111100000
3730 DATA 0001111111110000
3740 DATA 0000111011110000
3750 DATA 0000000111100000
3760 DATA 0000011111000000
3770 DATA 0000011000000000
3780 DATA 0000010000000000
3790 DATA 0000000000000000
3800 DATA 0000000000000000
3810 DATA 0000000000000000
3820 DATA 0000000000000000
3830 DATA 0000000000000000
3840 DATA 0000000000000000
3850 DATA 0000000000000000
3860 DATA 0000000000000000
3870 REM Sprite 9
3880 DATA 0000111111110000
3890 DATA 0000111111110000
3900 DATA 0000011111100000
3910 DATA 0000011111100000
3920 DATA 0000011101100000
3930 DATA 0000011000000000
3940 DATA 0000110000000000
3950 DATA 0000011000000000
3960 DATA 0000000000000000
3970 DATA 0000000000000000
3980 DATA 0000000000000000
3990 DATA 0000000000000000
4000 DATA 0000000000000000
4010 DATA 0000000000000000
4020 DATA 0000000000000000
4030 DATA 0000000000000000
4040 REM Sprite 25
4050 DATA 1111111100000010
4060 DATA 0001111111111001
4070 DATA 0000000000000100
4080 DATA 0000000000010110
4090 DATA 0000000000000110
4100 DATA 0000000000000000
4110 DATA 0000000000000001
4120 DATA 0000000000000000
```

```
4130 DATA 0000000000000000
4140 DATA 0000000000000000
4150 DATA 0000000000000000
4160 DATA 0000000000000000
4170 DATA 0000000000000000
4180 DATA 0000000000000000
4190 DATA 0000000000000000
4200 DATA 0000000000000000
4210 REM Sprite 10 (zwart)
4220 DATA 0111111111011110
4230 DATA 0100111111110010
4240 DATA 0000001111000000
4250 DATA 0000011111100000
4260 DATA 0000001111000000
4270 DATA 0000001111000000
4280 DATA 0000001111000000
4290 DATA 0000000000000000
4300 DATA 0000000000000000
4310 DATA 0000000000000000
4320 DATA 0000000000000000
4330 DATA 0000000000000000
4340 DATA 0000000000000000
4350 DATA 0000000000000000
4360 DATA 0000000000000000
4370 DATA 0000000000000000
4380 REM Sprite 11 (rood)
4390 DATA 0000000000000000
4400 DATA 0011000000001100
4410 DATA 0011000000001100
4420 DATA 0000000000000000
4430 DATA 0000010000100000
4440 DATA 0000010000100000
4450 DATA 0000000000000000
4460 DATA 0000000000000000
4470 DATA 0000001111000000
4480 DATA 0000001111000000
4490 DATA 0000011111100000
4500 DATA 0000011111100000
4510 DATA 0000010110100000
4520 DATA 0000010110100000
4530 DATA 0000011111100000
4540 DATA 0000000000000000
4550 REM Sprite 12 (lichtblauw)
4560 DATA 0000000000000000
4570 DATA 0000000000000000
4580 DATA 0000000000000000
4590 DATA 0011000000001100
4600 DATA 0011000000001100
4610 DATA 0011000000001100
```

```
4620 DATA 0011001111001100
4630 DATA 0011101111011100
4640 DATA 0001110000111000
4650 DATA 0001110000111000
4660 DATA 0000100000010000
4670 DATA 0000100000010000
4680 DATA 0000101001010000
4690 DATA 0000001001000000
4700 DATA 0000000000000000
4710 DATA 0000000000000000
4720 REM Sprite 22
4730 DATA 1111111111111111
4740 DATA 1001000110001001
4750 DATA 1010101111010101
4760 DATA 1100010110100011
4770 DATA 0111111111111110
4780 DATA 0000000110000000
4790 DATA 0000000000000000
4800 DATA 0000000000000000
4810 DATA 0000000000000000
4820 DATA 0000000000000000
4830 DATA 0000000000000000
4840 DATA 0000000000000000
4850 DATA 0000000000000000
4860 DATA 0000000000000000
4870 DATA 0000000000000000
4880 DATA 0000000000000000
4890 REM Sprite 23
4900 DATA 0000000000000000
4910 DATA 0000000000000000
4920 DATA 0000000000000000
4930 DATA 0000000000000000
4940 DATA 0000000000000000
4950 DATA 0000000001111111
4960 DATA 0000110011111110
4970 DATA 0001110111111100
4980 DATA 0010111111111000
4990 DATA 0111111111111100
5000 DATA 0001110111111011
5010 DATA 0000000000000000
5020 DATA 0000000000000000
5030 DATA 0000000000000000
5040 DATA 0000000000000000
5050 DATA 0000000000000000
5060 REM Sprite 24
5070 DATA 0000000000000000
5080 DATA 0000000000000000
5090 DATA 0000000000000000
5100 DATA 0000000000000000
```

```
5110 DATA 0000000000000000
5120 DATA 0000000000000000
5130 DATA 0000110000000000
5140 DATA 0001110111111000
5150 DATA 0010111111111100
5160 DATA 0111111111111110
5170 DATA 0011110111111111
5180 DATA 0000000000011111
5190 DATA 0000000000000000
5200 DATA 0000000000000000
5210 DATA 0000000000000000
5220 DATA 0000000000000000
5230 REM Sprite 25
5240 DATA 0001000000000000
5250 DATA 0010100000000000
5260 DATA 0111010000000000
5270 DATA 1111101000000000
5280 DATA 1111011000000000
5290 DATA 0111110000000000
5300 DATA 0011100000000000
5310 DATA 0000000000000000
5320 DATA 0000000000000000
5330 DATA 0000000000000000
5340 DATA 0000000000000000
5350 DATA 0000000000000000
5360 DATA 0000000000000000
5370 DATA 0000000000000000
5380 DATA 0000000000000000
5390 DATA 0000000000000000
5400 REM Achtergrondgrafiek
5410 IF JJK$="J" THEN 5430 ELSE COLOR 1,3,2:CLS
5420 REM Liftkabels
5430 FOR T=10 TO 182 STEP 2
5440 PSET(48,T),1
5450 NEXT T
5460 FOR T=9 TO 181 STEP 2
5470 PSET(47,T),1
5480 NEXT T
5490 FOR T=10 TO 182 STEP 2
5500 PSET(74,T),1
5510 NEXT T
5520 FOR T=9 TO 181 STEP 2
5530 PSET(73,T),1
5540 NEXT T
5550 FOR T=10 TO 182 STEP 2
5560 PSET(100,T),1
5570 NEXT T
5580 FOR T=9 TO 181 STEP 2
5590 PSET(99,T),1
```

```
5600 NEXT T
5610 IF JJK$="J" THEN JJK$="":RETURN
5620 LINE(74-9,184)-(74+9,192),1,BF
5630 LINE(74-9,0)-(74+9,8),1,BF
5640 LINE(48-9,184)-(48+9,192),1,BF
5650 LINE(48-9,0)-(48+9,8),1,BF
5660 LINE(100-9,184)-(100+9,192),1,BF
5670 LINE(100-9,0)-(100+9,8),1,BF
5680 FOR T=1 TO 192 STEP 5
5690 LINE(195,T)-(205,T+5),11,B
5700 NEXT T
5710 DRAW"bm0,60;c12r30"
5720 DRAW"bm5,160;c12r25"
5730 DRAW"bm118,160;c12r15f5r2e5r20f5e5r12"
5740 DRAW"bm118,60;c12r20f5r2e5r37"
5750 DRAW"bm135,110;c12r25f5e5r17"
5760 DRAW"bm215,60;c12r50"
5770 DRAW"bm215,110;c12r25"
5780 DRAW"bm215,160;c12r25"
5790 DRAW"bm0,20;c2f3e2r1f3r2f2e3
f2r2e3r2f2e2f1e4h3u2h2e3u2erh2e4"
5800 PAINT(10,10),2
5810
DRAW"bm0,61;c2r30f2d2g3d2l2g2f3r3f3g3f2g4d1f2r2f2d2g3d1g3l2
h2g3f3d2f2r2f2d2g2f3g5d2l1h3l1g3l2h2g4l3h2g3l1d2f3d2f1d2g3
d1g2d1f3d1f3d2g3d4r26f2g3d2l1g2f4d2g2f3d2g2f3d1f1d1"
5820 PAINT(10,100),2
5830 DRAW"bm115,0;c2f3d2f2g3d2g2d2f1g2f4r4e2r1f3r1e3u2e3r2
f3g1f2r2r2e3r1f2e3h2e3r2f2r3f2e3r2f3g2f2e3u2e2r3d2g2f4r2e3
r2u2e1h3u2e3u1e2h2e3"
5840 PAINT(150,5),2
5850 DRAW"bm215,0;c2f3d2g2d1g1d2f2g2f4r2f2r2e3h2e3f2r1f2e3
r2f3r2f2r2r2e2f3e2f1"
5860 PAINT(250,5),2
5870
DRAW"bm212,192;e3u2e2u1h2u2e3h2u2h3e2h3u2e2r27u2e2u3e2u2e2
u3h2u1e2u2e3u2e2u1h3l2h3l2g2h3l1h2g3l2g2l2h3g3hu1h2u2e2u1
h4u1e1r25u2e2u3e3u2e3u3h3u2h2u2e2h3l2h3g2h3e2h3g3l2g2l2h2
g3h3u2e3u1h3e2h1r50"
5880 PAINT(250,185),2
5890 DRAW"bm115,192;c2e3u2e3u1h3u2e2h3l2h1u2e3h2u1e3r15f5
r2e5r20f5e5r12f2d2g2d2g3l1h2g3d2f3d2f3r2d2f3d2f1"
5900 PAINT(160,170),2
5910 DRAW"bm187,61;c2l37g5l2h5l20df2r2f2d3g3l2g3d2f3g2d2f
3g2d2g2f3d2g4f3e2f3d3g3d2g3d3f3r2e3r1f2r2f3e2h2e1r3f3r2f4
e3u1e2r1f2g1f2r1e3r2e2r2f3r1e2r2f2r1f2r2e3u1e3u2e2u2e1h1
l18g5h5l25u2h2u1h3u2e1u2e2u1h2u2h1e2u2e3u2e3u1h2e2r2f3r2
f2r2e2r1f2r2e3r1f2r2f2r1e2r1
5920 DRAW"bm187,61;c2f2d4g3d2f2d1g3l2h3e2h3l2g3f2g2h2l1g2"
```

```
5930 PAINT (126,63),2
5940 DRAW"bm227,159;c1r812u114r1u813d1u2r3u2r1u213d1u1r1u1
r4f2d7r1d1r2e1g112u111u211u613d13r211u11"
5950 LINE(0,182)-(256,192),1,BF
5960 DRAW"bm 3,185"
5970 TIJD=5000
5980 COLOR 15
5990 PRINT #1,"Man:    Tijd:";
6000 PRINT #1,USING "####";TIJD;
6010 PRINT #1," Water"
6020 COLOR 1
6030 DI$="c15h4u1e2r4f2d1g3h1u111h111r1u1e1f1e1f1d1r111g111"
6040 VFL=6
6050 AR$="GH"
6060 COLOR 15
6070 GOSUB 6270
6080 COLOR 1
6090 RETURN
6100 REM Vlucht van spook
6110 DATA "U",9,"L",6,"R",6,"U",5,"L",
19,"R",3,"O",10,"R",16,"U",14,"E",0
6120 RESTORE 6110
6130 FOR T=1 TO 10
6140 READ H$(T),H(T)
6150 NEXT T
6160 REM Posities van diamanten
6170 DATA 15,159,125,159,146,109,230,109,130,59
6180 RESTORE 6170
6190 FOR T=1 TO 5
6200 READ PX(T),PY(T)
6210 NEXT T
6220 FOR T=1 TO 5
6230 P$(T)="N"
6240 NEXT T
6250 RETURN
6260 REM Mannetje verloren
6270 VFL=VFL-1
6280 IF VFL=0 THEN LINE(35,185)-(46,195),1,BF:DRAW"bm 30,185":
PRINT #1,VFL:LINE(90,94)-(90+16*6,106),3,BF:DRAW"bm 90,96":
PRINT #1," GAME OVER  ":INTERVAL OFF:GOTO 7010
6290 LINE(35,185)-(46,192),1,BF
6300 DRAW"bm 30,185"
6310 PRINT #1,VFL
6320 IF AR$="GH" THEN AR$="":RETURN
6330 GOTO 400
6340 RETURN
6350 REM Tanken
6360 IF X>220 AND X<230 AND Y>130 AND Y<140 THEN
 GOSUB 6700:RETURN
```

```
ELSE SCHOT=1:RETURN
6370 REM Schieten
6380 IF LE$="J" THEN RETURN
6390 IF WA$="LEEG" THEN RETURN
6400 IF R$="r" THEN PUT SPRITE 20,(X+16,Y+10),4,10:
GOSUB 6450 ELSE PUT
SPRITE 20,(X-16,Y+10),4,20:GOSUB 6470
6410 SCHOT=0
6420 GOSUB 6660
6430 PUT SPRITE 20,(0,0),0,31
6440 RETURN
6450 IF O>X+14 AND O<X+33 AND P>Y+8 AND P<Y+19 THEN
 L=1:L1=0:O=194:P=-1:TIJD=TIJD+100:RETURN
6460 RETURN
6470 IF O>X-17 AND O<X+2 AND P>Y+8 AND P<Y+19 THEN
 L=1:L1=0:O=194:P=-1:TIJD=TIJD+100:RETURN
6480 RETURN
6490 SCHOT=0
6500 RETURN
6510 REM Liften stoppen
6520 LI$="N"
6530 TR$=STR$(A1)
6540 DRAW "bm40,"+TR$+";c1"+Q$
6550 PUT SPRITE 10,(0,0),15,31
6560 TR$=STR$(A3)
6570 DRAW "bm92,"+TR$+";c1"+Q$
6580 PUT SPRITE 12,(0,0),15,31
6590 TR$=STR$(A4)
6600 DRAW "bm40,"+TR$+";c1"+Q$
6610 PUT SPRITE 13,(0,0),15,31
6620 TR$=STR$(A6)
6630 DRAW "bm92,"+TR$+";c1"+Q$
6640 PUT SPRITE 15,(255,191),15,31
6650 RETURN
6660 IF WA=51 THEN WA$="LEEG":RETURN
6670 LINE(248-WA,186)-(248,188),1,BF
6680 WA=WA+1
6690 RETURN
6700 IF WA$<>"LEEG" THEN RETURN
6710 WA$=""
6720 FOR T=1 TO 50
6730 PSET(248-T,186),2
6740 PSET(248-T,187),7
6750 PSET(248-T,188),2
6760 NEXT T
6770 WA=0
6780 RETURN
6790 TIJD=TIJD-100
6800 IF TIJD<=0 THEN 6270 ELSE LINE(96,185)-(128,192),1,BF:
```

```
DRAW"bm 100,185":COLOR 15:PRINT #1,USING"####";TIJD:RETURN
6810 REM Titelscherm
6820 SCREEN 2
6830 COLOR 15,1,1
6840 CLS
6850 DRAW"bm60,40;c15d30u30f10e10d30"
6860 DRAW"bm90,70;c15u30r15d15l15"
6870 LINE (90,55)-(105,70),15
6880 DRAW"bm80,90;d30u30f10e10d30"
6890 DRAW"bm110,90;d30"
6900 DRAW"bm120,90;d30"
6910 DRAW"bm140,90;d30"
6920 LINE (120,90)-(140,120),15
6930 DRAW"bm150,90;r20l20d15r10l10d15r20"
6940 DRAW"bm180,120;u30r15d15l15"
6950 LINE (180,105)-(195,120),15
6960 DRAW"bm 5,160"
6970 PRINT #1,"<c> by ARNE & DANIEL STOFFREGEN"
6980 FOR T=1 TO 1000
6990 NEXT T
7000 RETURN
7010 FOR T=1 TO 2000
7020 NEXT T
7030 COLOR 15,1,1
7040 SCREEN 0
7050 LOCATE 10,12,0
7060 PRINT"Nog een spelletje?"
7070 LOCATE 10,15
7080 PRINT"Druk op ENTER"
7090 A$=INKEY$:IF A$="" THEN 7090
7100 IF A$=CHR$(13) THEN RUN ELSE 7090
```

9.17 Drie machinetaalroutines

Met de MSX kunt u naast MSX-BASIC ook programmeren in assembler. Assembler is een vertaalroutine van BASIC naar (hexadecimale) machinetaal. Omdat er in dit boek geen assembleer-programma staat afgedrukt, moet u de hexadecimale getallen zelf intikken. In de listings zijn ze van een uitgebreid commentaar voorzien. Eerst een korte beschrijving van de machinetaalroutines die als voorbeeld dienen.

- Met machinetaalroutine 1 krijgt u enigszins een indruk van de snelheid die u in machinetaal kunt bereiken. Het hele tekstscherm staat in een fractie van een seconde vol met een en hetzelfde karakter, de letter m. Probeer het programma zo

te veranderen dat u naar eigen inzicht een deel van het scherm kunt vullen met een willekeurig teken. Als u de toelichting in de listing goed bekijkt, zult u zien dat u alle kanten opkunt met deze routine.

- Machinetaalroutine 2 tovert u voor hoe u vliegensvlug een deel van het VRAM naar het RAM kopieert. In dit voorbeeld wordt het hele VRAM gekopieerd. Is dat gebeurd, dan kunt u met een gerust geweten terugkeren naar het tekstscherm want...
- ... met machinetaalroutine 3 kunt u het scherm dat u zojuist hebt opgeslagen in het RAM weer terugsturen naar het VRAM. Deze routine demonstreert hoe u een grafische afbeelding maakt, die opslaat met machinetaalroutine 2, een tweede plaatje maakt en intussen de eerste afbeelding weer terugroept. De enige beperking is de geheugencapaciteit van het RAM. Als u zich bij het kopiëren beperkt tot delen van grafische plaatjes, zijn de toepassingsmogelijkheden onbeperkt.

Machinetaalroutine 1

```
10 REM Demonstratie van het gebruik van ROM-routines
20 REM &H0056: een deel vam het VRAM vullen met een teken
30 CLEAR 200,39999!
40 SCREEN 0
50 COLOR 1,15
60 WIDTH 40
70 FOR N=40000! TO 40011!
80 READ A$
90 POKE N,VAL("&H"+A$)
100 NEXT N
110 REM &H4D=77 is de letter 'm' (dec. 77)
120 DATA 3e,4d
130 REM &H00 en &H00 geven aan waar in het VRAM moet worden
 begonnen
140 DATA 21,00,00
150 REM &HC0 en &H03 geven aan tot waar het geheugen moet
 worden gevuld (dec. 960)
160 DATA 01,c0,03
170 REM &H56 en &H00 vormen het startadres van de ROM-routine
 (dec. 86)
180 DATA cd,56,00
190 REM Terug naar BASIC
200 DATA c9
210 DEFUSR=40000!
220 A=USR(0)
```

Machinetaalroutine 2

```
10 REM Demonstratie van het gebruik van ROM-routines
20 REM &H0059: een deel van het VRAM kopiëren naar RAM
30 CLEAR 200,39999!
40 SCREEN 0
50 COLOR 1,15
60 WIDTH 40
70 FOR N=40000! TO 40012!
80 READ A$
90 POKE N,VAL("&H"+A$)
100 NEXT N
110 REM &H00 en &H00 geven het eerste VRAM-adres aan dat moet
 worden gekopieerd (dec. 0)
120 DATA 21,00,00
130 REM &H00 en &H40 staan voor de lengte van het te kopiëren
 blok
140 DATA 01,00,40 'bc
150 REM &H28 en &Ha0 geven het beginadres van het RAM aan
160 DATA 11,28,a0 'de
170 REM &H59 en &H00 staan voor de transferroutine uit het ROM
 (dec. 89)
180 DATA cd,59,00 'call
190 REM Terug naar BASIC
200 DATA c9
210 DEFUSR=40000!
220 A=USR(0)
```

Machinetaalroutine 3

```
220 REM Tekening maken en opslaan in het RAM
230 REM Een transferroutine die de RAM-gegevens terugbrengt
 naar het VRAM
240 SCREEN 2
250 FOR N=10 TO 80 STEP 5
260 CIRCLE(128,96),N
270 NEXT N
280 FOR N=132 TO 255 STEP 10
290 PAINT(N,96)
300 NEXT N
310 A=USR(0)
320 CLS
330 FOR N=40000! TO 40012!
340 READ A$
350 POKE N,VAL("&h"+A$)
360 NEXT N
370 REM &H28 en &HA0 geven het eerste RAM-adres aan dat naar het
 VRAM moet worden gekopieerd (dec. 41000)
380 DATA 21,28,a0 'hl
390 REM &H00 ten &H40 geven de lengte aan van het te kopiëren
 blok (dec. 16384)
400 DATA 01,00,40 'bc
410 REM &H00 en &H00 geven het eerste VRAM-adres aan waar de
 RAM-waarden naartoe moeten (dec. 0)
420 DATA 11,00,00 'de
430 REM &H5c en &H00 staan voor de tranferroutines uit het ROM
 (dec. 92)
440 DATA cd,5c,00 'call
450 REM Terug naar BASIC
460 DATA c9
470 DEFUSR=40000!
480 CLS
490 FOR N=255 TO 10 STEP-10
500 LINE (0,0)-(N,N),INT(RND(1)*14+1),BF
510 NEXT N
520 A=USR(0)
530 GOTO 530
```

Bijlage 1 Belangrijke adressen in RAM en ROM

&H1BBF - &H1BBF+2047

De karakterset die bij het aanzetten van de computer uit het ROM wordt gekopieerd naar het VRAM. In bijlage 3.1 staat de hele karakterset afgebeeld. Het beginadres van deze karakterset bevindt zich bovendien op de geheugenplaatsen &HF920 en &HF921.

&HF3AE

Breedte van het beeldscherm (aantal karakters per regel) in SCREEN 0. Bij inschakeling van de MSX is dat 37 karakters per regel. Een verandering van deze waarde heeft pas effect als u opnieuw SCREEN 0 invoert.

&HF3AF

Breedte van het beeldscherm (aantal karakters per regel) in SCREEN 1. Bij inschakeling van de MSX is die 29 karakters per regel. Een verandering van deze waarde heeft pas effect als u opnieuw SCREEN 1 invoert.

&HF3B0

Breedte van het beeldscherm (aantal karakters per regel) in het actuele SCREEN. Een verandering van deze waarde treedt gelijk in op de regel onder de actuele cursorpositie. In tegenstelling tot de instructie WIDTH wordt daarbij het beeldscherm niet gewist.

&HF3B1

Aantal regels per (scherm)pagina. Het standaardaantal is 24. U kunt deze waarde naar believen veranderen. Een nieuwe SCREEN-instructie heeft hierop geen invloed. De ingevoerde waarde geldt voor de beide tekstschermen.

&HF3B3 en &HF3B4

komen overeen met BASE(0): name table - SCREEN 0

&HF3B7 en &HF3B8

komen overeen met BASE(2): pattern generator table - SCREEN 0

&HF3BD en &HF3BE

komen overeen met BASE(5): name table - SCREEN 1

&HF3BF en &HF3C0

komen overeen met BASE(6): colour table - SCREEN 1

&HF3C1 en &HF3C2

komen overeen met BASE(7): pattern generator table - SCREEN 1

&HF3C3 en &HF3C4

komen overeen met BASE(8): sprite attribute table - SCREEN 1

&HF3C5 en &HF3C6

komen overeen met BASE(9): sprite pattern table - SCREEN 1

&HF3C7 en &HF3C8

komen overeen met BASE(10): name table - SCREEN 2

&HF3C9 en &HF3CA

komen overeen met BASE(11): colour table - SCREEN 2

&HF3CB en &HF3CC

komen overeen met BASE(12): pattern generator table - SCREEN 2

&HF3CD en &HF3CE

komen overeen met BASE(13): sprite attribute table - SCREEN 2

&HF3CF en &HF3D0

komen overeen met BASE(14): sprite pattern table - SCREEN 2

&HF3D1 en &HF3D2

komen overeen met BASE(15): name table - SCREEN 3

&HF3D5 en &HF3D6

komen overeen met BASE(17): pattern generator table - SCREEN 3

&HF3D7 en &HF3D8

komen overeen met BASE(18): sprite attribute table - SCREEN 3

&HF3D9 en &HF3DA

komen overeen met BASE(19): sprite pattern table - SCREEN 3

&HF3DB

Toetsklik (instelbaar met de derde parameter van SCREEN). Als u de toetsklik hebt uitgeschakeld, staat hier een 0, als u die hebt ingeschakeld een 1. Deze functie kunt u natuurlijk ook met POKE in- of uitschakelen.

&HF3DC

Actuele positie van de Y-coördinaat van de cursor (dus van de regel waarop de cursor staat). Dit adres kunt u net zoals de instructie LOCATE gebruiken om de cursor te verplaatsen.

&HF3DD

Actuele positie van de X-coördinaat van de cursor (dus van de kolom waarin de cursor staat). Dit adres kunt u net zoals de instructie LOCATE gebruiken om de cursor te verplaatsen.

&HF3DE

De waarde op dit adres bepaalt of de functietoetsen al dan niet op het scherm komen te staan (functiebalk). Dit komt overeen met de functie KEY ON/KEY OFF. De waarde 255 betekent dat de functiebalk

op het scherm staat en de waarde 0 betekent dat die niet te zien is. POKEt u een andere waarde in dit adres, dan geldt de verandering pas wanneer u met CLS het scherm wist.

&HF3DF

komt overeen met de actuele inhoud van register VDP(0)

&HF3DF

komt overeen met de actuele inhoud van register VDP(1)

&HF3E1

komt overeen met de actuele inhoud van register VDP(2)

&HF3E2

komt overeen met de actuele inhoud van register VDP(3)

&HF3E3

komt overeen met de actuele inhoud van register VDP(4)

&HF3E4

komt overeen met de actuele inhoud van register VDP(5)

&HF3E5

komt overeen met de actuele inhoud van register VDP(6)

&HF3E6

komt overeen met de actuele inhoud van register VDP(7)

&HF3E7

komt overeen met de actuele inhoud van register VDP(8)

&HF3E9

Actuele voorgrondkleur (schrijf-/tekenkleur). Als u een andere waarde in dit adres POKEt, treedt de verandering pas in na een nieuwe SCREEN-instructie.

&HF3EA

Actuele achtergrondkleur. Als u een andere waarde in dit adres POKEt, treedt de verandering pas in na een nieuwe SCREEN-instructie.

&HF3EB

Actuele randkleur (geldt niet voor SCREEN 0). Als u een andere waarde in dit adres POKEt, treedt de verandering pas in na een nieuwe SCREEN-instructie.

&HF415

Actuele positie van printerkop/margrietwiel. De instructie LPOS verschaft u dezelfde informatie.

&HF417

Deze waarde komt overeen met parameter 5 van de instructie SCREEN: is een aangesloten printer een MSX-printer (waarde 0) of niet (waarde ongelijk 0)? In het laatste geval worden de speciale grafische karakters van de MSX uitgevoerd als spaties.

&HF87F - &HF87F+159

Deze 160 adressen bevatten de actuele inhoud van de functietoetsen. U kunt die inhoud opvragen met PEEK en veranderen met POKE. Veranderingen van toetsen worden pas zichtbaar als u de spatiebalk indrukt. Wilt u direct de nieuwe inhoud van een toets kunnen bekijken, dan moet u met VPOKE de nieuwe waarde in het VRAM POKEn. Bij de tekstbeeldschermen kunt u bovendien de actuele inhoud van de functietoetsen oproepen met VPEEK (daarbij worden, afhankelijk van de beeldschermbreedte, zoveel mogelijk karakters van de inhoud van de functietoets op het scherm gezet).

HF922 en &HF923

Het actuele beginadres van de name table. Net als bij de instructie VDP kunt u dit adres niet alleen lezen (dat komt overeen met een BASE-waarde), maar ook rechtstreeks veranderen met POKE. Verander dit adres pas als u zeker weet dat dat geen schade kan opleveren (zie ook SCREEN, VDP en BASE).

HF92 en &HF925

Het actuele beginadres van de pattern generator table. Net als bij de instructie VDP kunt u dit adres niet alleen lezen (dat komt overeen met een BASE-waarde), maar ook rechtstreeks veranderen met POKE. Verander dit adres pas als u zeker weet dat dat geen schade kan opleveren (zie ook SCREEN, VDP en BASE).

HF926 en &HF927

Het actuele beginadres van de sprite pattern table. Net als bij de instructie VDP kunt u dit adres niet alleen lezen (dat komt overeen met een BASE-waarde), maar ook rechtstreeks veranderen met POKE. Verander dit adres pas als u zeker weet dat dat geen schade kan opleveren (zie ook SCREEN, VDP en BASE).

HF928 en &HF929

Het actuele beginadres van de sprite attribute table. Net als bij de instructie VDP kunt u dit adres niet alleen lezen (dat komt overeen met een BASE-waarde), maar ook rechtstreeks veranderen met POKE. Verander dit adres pas als u zeker weet dat dat geen schade kan opleveren (zie ook SCREEN, VDP en BASE).

&HF956 en verder

Geheugenblok waarin de noten worden opgeslagen die met de instructie PLAY worden ingevoerd. Vanuit dit blok leest de MSX via een interruptroutine de invoer toon voor toon in. Zie voor een gedetailleerde beschrijving van enkele speciale adressen paragraaf 6.2 (PLAY(N)).

&HBCC

Actuele karakter dat zich onder de cursor bevindt (invers weergegeven). Als de cursor naar een ander karakter wordt gedirigeerd, verandert hij van uiterlijk. De inhoud van dit adres verandert dus steeds, afhankelijk van de cursorpositie.

&HFC9C

Y-coördinaat van een aangesloten grafisch tableau (op te roepen met de instructie PAD).

&HFC9D

X-coördinaat van een aangesloten grafisch tableau (op te roepen met de instructie PAD).

&HFCA9

komt overeen met de waarde van parameter 3 van de instructie LOCATE: cursor tijdens de uitvoering van een programma zichtbaar (1) of niet zichtbaar (0) op het scherm? Die parameter kunt u dus ook via dit adres veranderen.

&HFCAB

Kleine letters/hoofdletters. Een 0 op dit adres betekent dat u met kleine letters en hoofdletters werkt (waarbij hoofdletters met SHIFT worden ingevoerd). Een 1 betekent dat u alleen met hoofdletters werkt, zonder dat u de CAPS-toets hebt gebruikt (de LED in de CAPS-toets is dan wel uitgeschakeld). U verandert de waarde met POKE.

&HFCAF

Actuele SCREEN.

&HFCB0

Laatst gekozen SCREEN. Als u dit adres oproept, kunt u ook zien op welk tekstscherm u het laatst hebt gewerkt, terwijl het actuele scherm een grafisch scherm is.

&HFCB2

In SCREEN 3 hebt u de mogelijkheid bij de instructie PAINT niet alleen te bepalen met welke kleur wordt getekend, maar ook tot aan welke kleur moet worden getekend. Die laatste kleur staat op dit adres.

&HFCB3

Actuele X-coördinaat van de grafische cursor.

&HFCB5

Actuele Y-coördinaat van de grafische cursor.

&HFCB7 en &HFCB9

Het verschil tussen de relatieve en de absolute grafische cursorpositie. Dit verschil doet zich voor bij relatieve instructies zoals DRAW"N", STEP en bij programma's waarbij sprake is van een middelpunt van een cirkel en cirkelpunten).

&HFCBC

Schaalgrootte. Bij de instructie DRAW kunt u met de parameter S, gevolgd door een getal, de schaal van een tekening bepalen. De actuele vergroting/verkleining (de standaardwaarde is 4 en komt overeen met een schaal van 1:1) is opgeslagen in dit adres. Met POKE kunt u de schaal natuurlijk ook rechtstreeks veranderen.

&HFCBD

Actuele draaiingshoek. Bij de instructie DRAW kunt u met de parameter A, gevolgd door een getal, in stappen van 90 graden tekenen (de standaardwaarde is 0 en betekent geen draaiing).

Bijlage 2 Mogelijke waarden voor de VDP-registers

Bijlage 2 Mogelijke waarden voor de VDP-registers

Waarden die we voor VDP-register 2 kunnen invoeren

00	0000	01	0400	02	0800	03	0C00
04	1000	05	1400	06	1800	07	1C00
08	2000	09	2400	0A	2800	0B	2C00
0C	3000	0D	3400	0E	3800	0F	3C00

Waarden die we voor VDP-register 4 kunnen invoeren

00	0000	01	0800	02	1000	03	1800
04	2000	05	2800	06	3000	07	3800

Waarden die we voor VDP-register 5 kunnen invoeren

00	0000	01	0080	02	0100	03	0180
04	0200	05	0280	06	0300	07	0380
08	0400	09	0480	0A	0500	0B	0580
0C	0600	0D	0680	0E	0700	0F	0780
10	0800	11	0880	12	0900	13	0980
14	0A00	15	0A80	16	0B00	17	0B80
18	0C00	19	0C80	1A	0D00	1B	0D80
1C	0E00	1D	0E80	1E	0F00	1F	0F80
20	1000	21	1080	22	1100	23	1180
24	1200	25	1280	26	1300	27	1380
28	1400	29	1480	2A	1500	2B	1580
2C	1600	2D	1680	2E	1700	2F	1780
30	1800	31	1880	32	1900	33	1980
34	1A00	35	1A80	36	1B00	37	1B80
38	1C00	39	1C80	3A	1D00	3B	1D80
3C	1E00	3D	1E80	3E	1F00	3F	1F80
40	2000	41	2080	42	2100	43	2180
44	2200	45	2280	46	2300	47	2380
48	2400	49	2480	4A	2500	4B	2580
4C	2600	4D	2680	4E	2700	4F	2780
50	2800	51	2880	52	2900	53	2980
54	2A00	55	2A80	56	2B00	57	2B80
58	2C00	59	2C80	5A	2D00	5B	2D80
5C	2E00	5D	2E80	5E	2F00	5F	2F80
60	3000	61	3080	62	3100	63	3180
64	3200	65	3280	66	3300	67	3380
68	3400	69	3480	6A	3500	6B	3580
6C	3600	6D	3680	6E	3700	6F	3780
70	3800	71	3880	72	3900	73	3980
74	3A00	75	3A80	76	3B00	77	3B80
78	3C00	79	3C80	7A	3D00	7B	3D80
7C	3E00	7D	3E80	7E	3F00	7F	3F80

Waarden die we voor VDP-register 6 kunnen invoeren

00	0000	01	0800	02	1000	03	1800
04	2000	05	2800	06	3000	07	3800

Waarden die we voor VDP-register 3 kunnen invoeren

deel 1

00	0000	01	0040	02	0080	03	00C0
04	0100	05	0140	06	0180	07	01C0
08	0200	09	0240	0A	0280	0B	02C0
0C	0300	0D	0340	0E	0380	0F	03C0
10	0400	11	0440	12	0480	13	04C0
14	0500	15	0540	16	0580	17	05C0
18	0600	19	0640	1A	0680	1B	06C0
1C	0700	1D	0740	1E	0780	1F	07C0
20	0800	21	0840	22	0880	23	08C0
24	0900	25	0940	26	0980	27	09C0
28	0A00	29	0A40	2A	0A80	2B	0AC0
2C	0B00	2D	0B40	2E	0B80	2F	0BC0
30	0C00	31	0C40	32	0C80	33	0CC0
34	0D00	35	0D40	36	0D80	37	0DC0
38	0E00	39	0E40	3A	0E80	3B	0EC0
3C	0F00	3D	0F40	3E	0F80	3F	0FC0
40	1000	41	1040	42	1080	43	10C0
44	1100	45	1140	46	1180	47	11C0
48	1200	49	1240	4A	1280	4B	12C0
4C	1300	4D	1340	4E	1380	4F	13C0
50	1400	51	1440	52	1480	53	14C0
54	1500	55	1540	56	1580	57	15C0
58	1600	59	1640	5A	1680	5B	16C0
5C	1700	5D	1740	5E	1780	5F	17C0
60	1800	61	1840	62	1880	63	18C0
64	1900	65	1940	66	1980	67	19C0
68	1A00	69	1A40	6A	1A80	6B	1AC0
6C	1B00	6D	1B40	6E	1B80	6F	1BC0
70	1C00	71	1C40	72	1C80	73	1CC0
74	1D00	75	1D40	76	1D80	77	1DC0
78	1E00	79	1E40	7A	1E80	7B	1EC0
7C	1F00	7D	1F40	7E	1F80	7F	1FC0

Waarden die we voor VDP-register 3 kunnen invoeren

deel 2

80	2000	81	2040	82	2080	83	20C0
84	2100	85	2140	86	2180	87	21C0
88	2200	89	2240	8A	2280	8B	22C0
8C	2300	8D	2340	8E	2380	8F	23C0
90	2400	91	2440	92	2480	93	24C0
94	2500	95	2540	96	2580	97	25C0
98	2600	99	2640	9A	2680	9B	26C0
9C	2700	9D	2740	9E	2780	9F	27C0
A0	2800	A1	2840	A2	2880	A3	28C0
A4	2900	A5	2940	A6	2980	A7	29C0
A8	2A00	A9	2A40	AA	2A80	AB	2AC0
AC	2B00	AD	2B40	AE	2B80	AF	2BC0
B0	2C00	B1	2C40	B2	2C80	B3	2CC0
B4	2D00	B5	2D40	B6	2D80	B7	2DC0
B8	2E00	B9	2E40	BA	2E80	BB	2EC0
BC	2F00	BD	2F40	BE	2F80	BF	2FC0
C0	3000	C1	3040	C2	3080	C3	30C0
C4	3100	C5	3140	C6	3180	C7	31C0
C8	3200	C9	3240	CA	3280	CB	32C0
CC	3300	CD	3340	CE	3380	CF	33C0
D0	3400	D1	3440	D2	3480	D3	34C0
D4	3500	D5	3540	D6	3580	D7	35C0
D8	3600	D9	3640	DA	3680	DB	36C0
DC	3700	DD	3740	DE	3780	DF	37C0
E0	3800	E1	3840	E2	3880	E3	38C0
E4	3900	E5	3940	E6	3980	E7	39C0
E8	3A00	E9	3A40	EA	3A80	EB	3AC0
EC	3B00	ED	3B40	EE	3B80	EF	3BC0
F0	3C00	F1	3C40	F2	3C80	F3	3CC0
F4	3D00	F5	3D40	F6	3D80	F7	3DC0
F8	3E00	F9	3E40	FA	3E80	FB	3EC0
FC	3F00	FD	3F40	FE	3F80	FF	3FC0

Bijlage 3 Karaktersets

Bijlage 3.1 De standaard karakterset

karakter 0



karakter 1



karakter 2



karakter 3



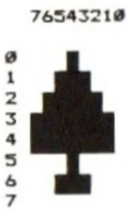
karakter 4



karakter 5



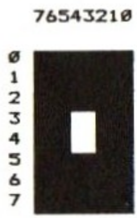
karakter 6



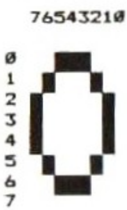
karakter 7



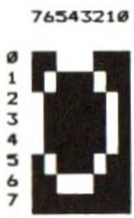
karakter 8



karakter 9



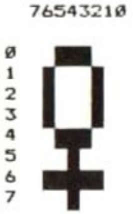
karakter 10



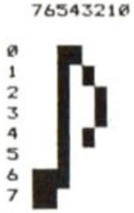
karakter 11



karakter 12



karakter 13



karakter 14



karakter 15



karakter 16



karakter 17



karakter 18



karakter 19



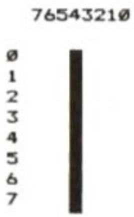
karakter 20



karakter 21



karakter 22



karakter 23





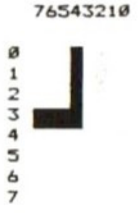
karakter 24



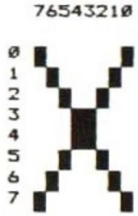
karakter 25



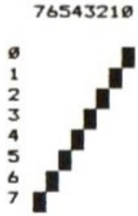
karakter 26



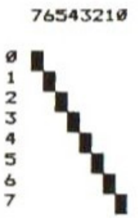
karakter 27



karakter 28



karakter 29



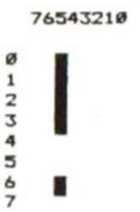
karakter 30



karakter 31



karakter 32



karakter 33

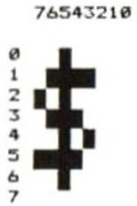


karakter 34

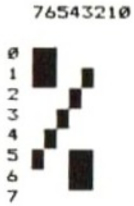


karakter 35

karakter 36



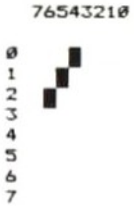
karakter 37



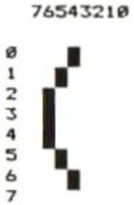
karakter 38



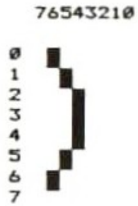
karakter 39



karakter 40



karakter 41



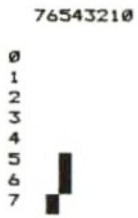
karakter 42



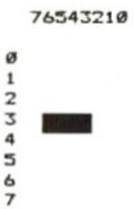
karakter 43



karakter 44



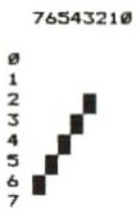
karakter 45



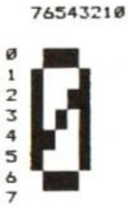
karakter 46



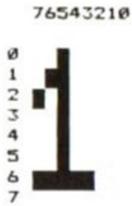
karakter 47



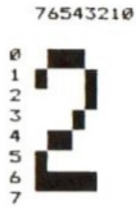
karakter 48



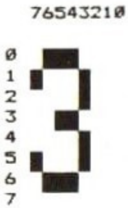
karakter 49



karakter 50



karakter 51



karakter 52



karakter 53



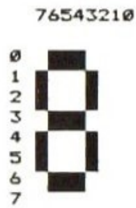
karakter 54



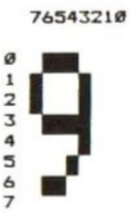
karakter 55



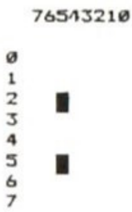
karakter 56



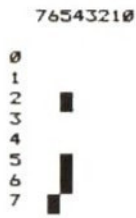
karakter 57



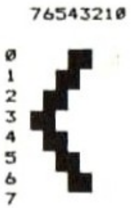
karakter 58



karakter 59



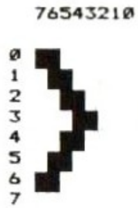
karakter 60



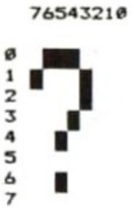
karakter 61



karakter 62



karakter 63



karakter 64



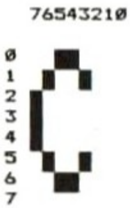
karakter 65



karakter 66



karakter 67



karakter 68



karakter 69



karakter 70

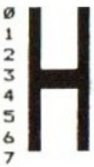


karakter 71



Char: karakter 72

76543210



karakter 73

76543210



karakter 74

76543210



karakter 75

76543210



Char karakter 76

76543210



karakter 77

76543210



karakter 78

76543210



Char karakter 79

76543210



karakter 80

76543210



karakter 81

76543210



karakter 82

76543210



karakter 83

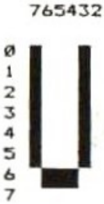
76543210



karakter 84



karakter 85



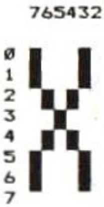
karakter 86



karakter 87



Char: karakter 88



karakter 89



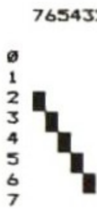
karakter 90



karakter 91



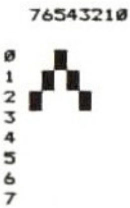
karakter 92



karakter 93



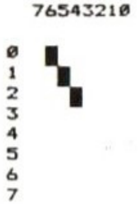
karakter 94



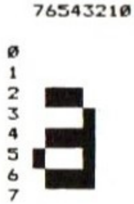
karakter 95



karakter 96



karakter 97



karakter 98



karakter 99



karakter 100



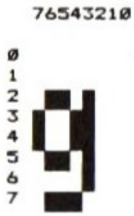
karakter 101



karakter 102



karakter 103



karakter 104



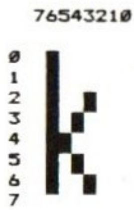
karakter 105



karakter 106



karakter 107



karakter 108



karakter 109



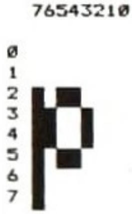
karakter 110



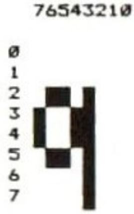
karakter 111



karakter 112



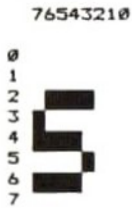
karakter 113



karakter 114



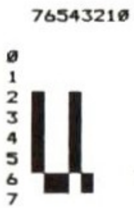
karakter 115



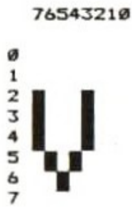
karakter 116



karakter 117



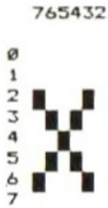
karakter 118



karakter 119



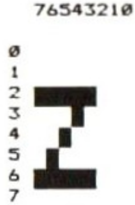
karakter 120



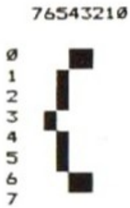
karakter 121



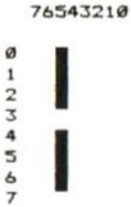
karakter 122



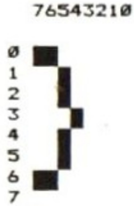
karakter 123



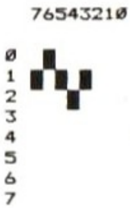
karakter 124



karakter 125



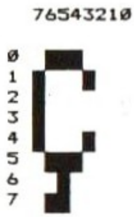
karakter 126



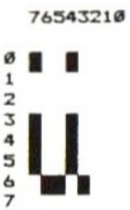
karakter 127



karakter 128



karakter 129



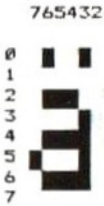
karakter 130



karakter 131



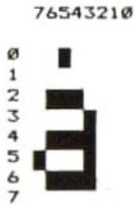
karakter 132



karakter 133



karakter 134



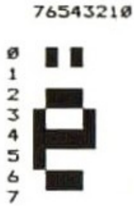
karakter 135



karakter 136



karakter 137



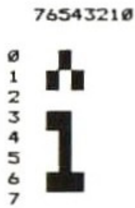
karakter 138



karakter 139



karakter 140



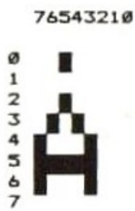
karakter 141



karakter 142



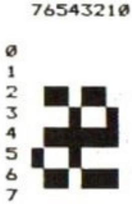
karakter 143



karakter 144



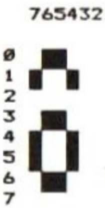
karakter 145



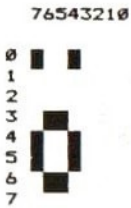
karakter 146



karakter 147



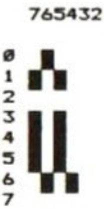
karakter 148



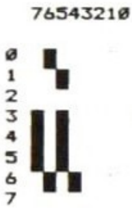
karakter 149



karakter 150



karakter 151



karakter 152



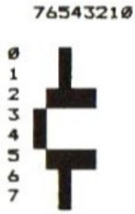
karakter 153



karakter 154



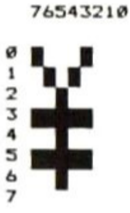
karakter 155



karakter 156



karakter 157



karakter 158



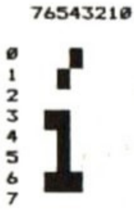
karakter 159



karakter 160



karakter 161



karakter 162



karakter 163



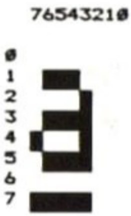
karakter 164



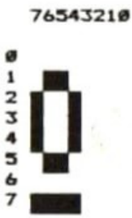
karakter 165



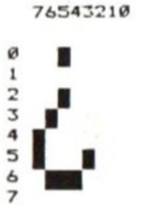
karakter 166



karakter 167



karakter 168



karakter 169



karakter 170



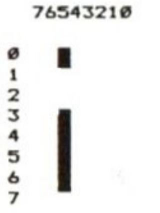
karakter 171



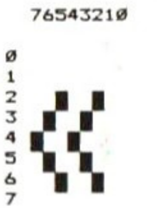
karakter 172



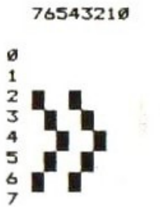
karakter 173



karakter 174



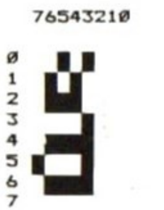
karakter 175



karakter 176



karakter 177



karakter 178



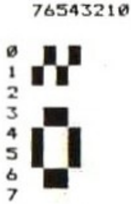
karakter 179



karakter 180



karakter 181



karakter 182



karakter 183



karakter 184



karakter 185



karakter 186



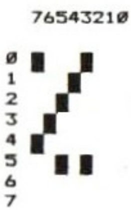
karakter 187



karakter 188



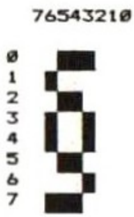
karakter 189



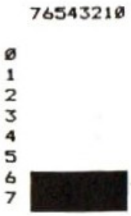
karakter 190



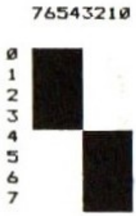
karakter 191



karakter 192



karakter 193



karakter 194



karakter 195



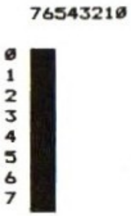
karakter 196



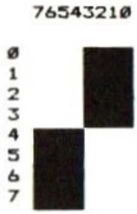
karakter 197



karakter 198



karakter 199



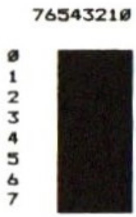
karakter 200



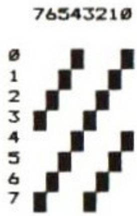
karakter 201



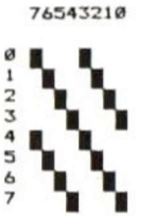
karakter 202



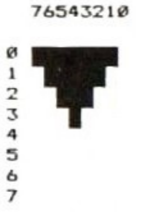
karakter 203



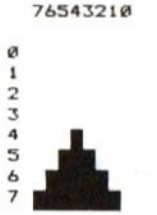
karakter 204



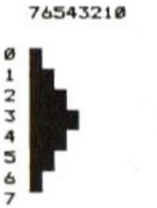
karakter 205



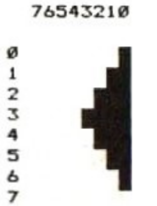
karakter 206



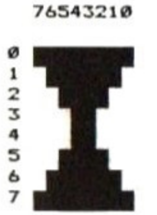
karakter 207



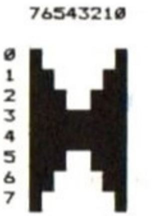
Char karakter 208



karakter 209



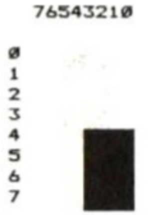
karakter 210



karakter 211



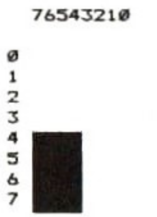
Char karakter 212



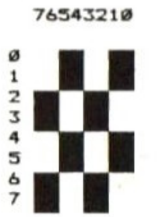
karakter 213



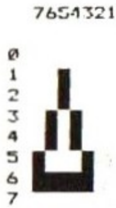
karakter 214



karakter 215



karakter 216



karakter 217



karakter 218



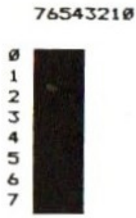
karakter 219



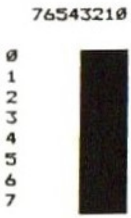
karakter 220



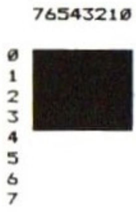
karakter 221



karakter 222



karakter 223



karakter 224



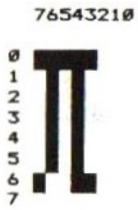
karakter 225



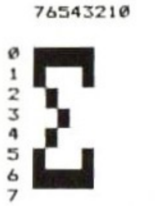
karakter 226



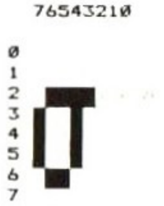
karakter 227



karakter 228



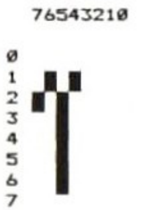
karakter 229



karakter 230



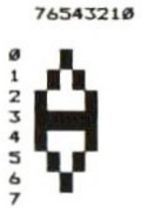
karakter 231



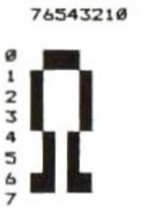
karakter 232



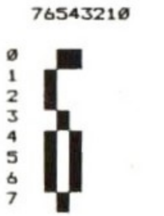
karakter 233



karakter 234



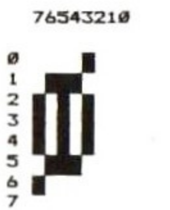
karakter 235



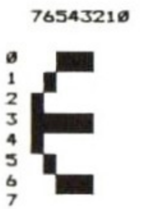
karakter 236



karakter 237



karakter 238



karakter 239



karakter 240

76543210



karakter 241

76543210



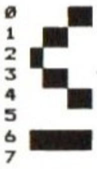
karakter 242

76543210



karakter 243

76543210



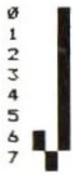
karakter 244

76543210



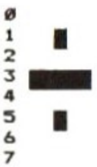
karakter 245

76543210



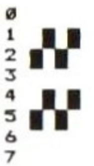
karakter 246

76543210



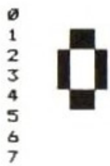
karakter 247 7

76543210



karakter 248

76543210



karakter 249

76543210



karakter 250

76543210



karakter 251

76543210



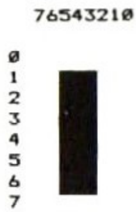
karakter 252



karakter 253



karakter 254



karakter 255

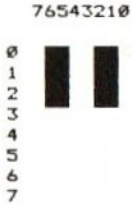


Bijlage 3.2 Extra MSX-karakterset

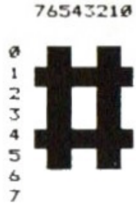
karakter 129



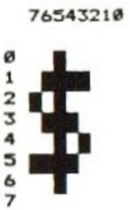
karakter 130



karakter 131



karakter 132



karakter 133



Char karakter 134



karakter 135



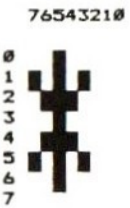
karakter 136



karakter 137



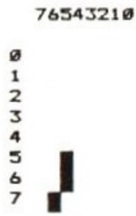
karakter 138



karakter 139



karakter 140



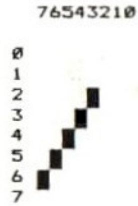
karakter 141



karakter 142



karakter 143



karakter 144



karakter 145



karakter 146



karakter 147



karakter 148



karakter 149



karakter 150



karakter 151



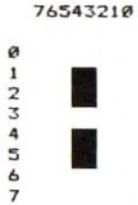
karakter 152



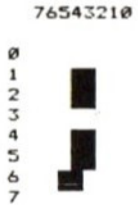
karakter 153



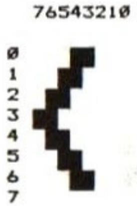
karakter 154



karakter 155



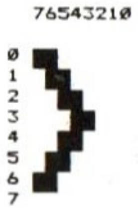
karakter 156



karakter 157



karakter 158



karakter 159



karakter 160



karakter 161



karakter 162



karakter 163



karakter 164



karakter 165



karakter 166



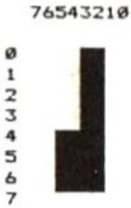
karakter 167



karakter 168



karakter 169



karakter 170



karakter 171



karakter 172



karakter 173



karakter 174



karakter 175



karakter 176



karakter 177



karakter 178



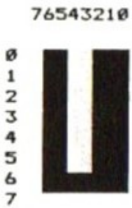
karakter 179



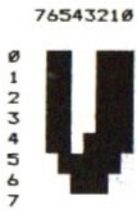
karakter 180



karakter 181



karakter 182



karakter 183



karakter 184



karakter 185



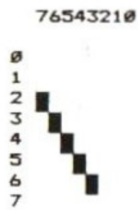
karakter 186



karakter 187



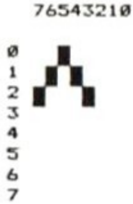
karakter 188



karakter 189



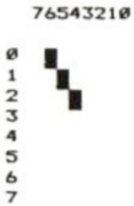
karakter 190



karakter 191



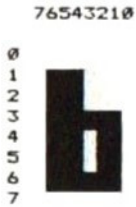
karakter 192



karakter 193



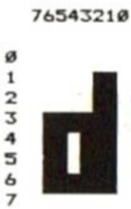
karakter 194



karakter 195



karakter 196



karakter 197



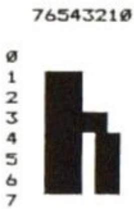
karakter 198



karakter 199



karakter 200



karakter 201

76543210

0 1 2 3 4 5 6 7



karakter 202

76543210

0 1 2 3 4 5 6 7



karakter 203

76543210

0 1 2 3 4 5 6 7



karakter 204

76543210

0 1 2 3 4 5 6 7



karakter 205

76543210

0 1 2 3 4 5 6 7



karakter 206

76543210

0 1 2 3 4 5 6 7



karakter 207

76543210

0 1 2 3 4 5 6 7



karakter 208

76543210

0 1 2 3 4 5 6 7



karakter 209

76543210

0 1 2 3 4 5 6 7



karakter 210

76543210

0 1 2 3 4 5 6 7



karakter 211

76543210

0 1 2 3 4 5 6 7



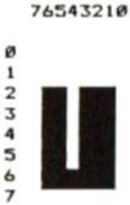
karakter 212

76543210

0 1 2 3 4 5 6 7



karakter 213



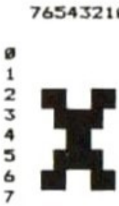
karakter 214



karakter 215



karakter 216



karakter 217



karakter 218





MSX Programma-verzameling

Een veelomvattende programmaverzameling: van een dis-assembler tot een sporttabel voor de standen in de eerste en ere-divisie van de voetbalcompetitie.
Ook bevat dit boek spelletjes en complete utilities.
Uit de inhoud: hexdump * grafiek editor * geluid * Umlauten op het scherm * crossreference voor variabelen * kalender * dis-assembler * gegevensverwerking/langspeelplatenbestand * balkdiagrammen-sporttabel ere- en eerste divisie voetbal-competitie.

ISBN 90 229 3350 4

MSX-computers hebben een streepje voor op alle andere computers: enerzijds ligt de verhouding prestatie-prijs zeer gunstig, anderzijds heeft deze computer buitengewone grafische- en geluids-mogelijkheden.
Wanneer u geïnteresseerd bent in deze mogelijkheden mag dit boek niet in uw MSX-bibliotheek ontbreken. Deze handleiding beschijft uitvoerig alle MSX-commando's voor het optimaal gebruik van de MSX-computer.

ISBN 90 229 3358 X

MSX Bibliotheek 2

DATA BECKER
NEDERLANDS *